МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой _____
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **2016 г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Исследование и анализ технических средств защиты авторских прав (DRM-защиты)

Специальность Вычислительная техника и программное обеспечение

Выполнил (а) Габдолов А.Н. ВТ-1Р-4
(Фамилия и инициалы) группа

Научный руководитель Муканова Т.Д., к.т.н., ст. преп.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекишева А.Ч., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «16» 05 2016 г.
(подпись)

по безопасности жизнедеятельности:
Прокофьева Н.Г., д.х.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «05» 05 2016 г.
(подпись)

по применению вычислительной техники:
Муканова Т.Д., к.т.н., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «16» 05 2016 г.
(подпись)

Нормоконтролер: Муканова Т.Д., к.т.н., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «16» 05 2016 г.
(подпись)

Рецензент: Усенбаева С.А., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20___ г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Экономических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ

на выполнение дипломного проекта

Студент Бобделв Айдос Назарханевич
(фамилия, имя, отчество)

Тема проекта Исследование и анализ технических средств защиты авторских прав (DRM-системы)

утверждена приказом ректора № 148 от «19» октября 2015 г.
Срок сдачи законченной работы « 30 » мая 2016 г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Исследование защиты интеллектуальной собственности с применением технических средств защиты авторских прав

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1. Понятие авторского права в наше время
2. DRM как средство защиты авторских прав
3. Изучение методов работы защиты
4. Способы взлома и противодействия
5. Взлом одной из популярных систем защиты на практике

Перечень графического материала (с точным указанием обязательных чертежей)

Иллюстрация принципа работы защиты

Изображения защищенных компакт-дисков

Иллюстрации процесса отладки

Рекомендуемая основная литература

1. Закон Республики Казахстан „Об авторском праве и смежных правах" от 10 июня 1996 года (с изменениями, внесенными Законами РК

2. Крис Касперски. Искусство дизассемблирования – СПб, БХВ-Петербург, 2008г.

3. Пилюгин Д.С. Реверсинг и защита программ от взлома – СПб, БХВ-Петербург, 2006г.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Приходько Н.Г.	29.02 – 05.05.16	[illegible]
Эконом. часть	Бекмухаметова А.И.	14.03 – 16.05.16	[illegible]
Примен ВТ	Мусатырова Г.Б	15.03 – 12.05.16	[illegible]
Нормоконтролер	Мусатырова Г.Б	26.05.16	[illegible]

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Постановка задачи	11.01.16 - 12.01.16	
2.	Разработка и утверждение технического задания (ТЗ)	13.01 - 20.01.16	
3.	Подбор и изучение материалов по тематике	21.01 - 12.02.16	
4.	Обзор актуальных систем DRM-защит.	15.02.16 - 4.03.16	
5.	Проведение взлома одной из популярнейших систем защит на практике	4.03.16 - 16.05.16	
6.	Расчет экономической части работы	14.03. - 16.05.16	
7.	Проведение расчетов, связанных с безопасностью жизнедеятельности.	29.02. - 05.05.16	
8.	Подведение итогов	16.05 - 26.05.16	

Дата выдачи задания «7» января 2016 г.

Заведующий кафедрой _____ Кумалбаев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Мухатирова Г.Ф.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент Гибадлов _____ Гибадлов А.Н.
(подпись) (Фамилия и инициалы)

Аннотация

В данной дипломной работе были рассмотрены актуальные средства защиты авторских прав (DRM-защиты). Изучены принцип работы, способы взлома и противодействия оному, выявлены сильные и слабые стороны, а также раскрыто негативное влияние данной защиты на добросовестных пользователей и систему в целом. Была рассмотрена целесообразность внедрения такой системы защиты в программные продукты.

Исследование в первую очередь направленно на повышение грамотности населения страны, а также повлияет на разработку будущих систем защиты и предложит правообладателем альтернативные способы защиты продукта.

Аңдатпа

Бұл дипломдық жұмыста авторлық құқықты қорғаудың (DRM-қорғау) өзекті тәсілдері қарастырылды. Жұмыс принципі, бұзу және бұзуға қарсы әдістері зерттелді, күшті және нашар жақтары анықталды, сондай-ақ бұл қорғаудың адал пайдаланушыларға және бүкіл жүйесінің жағымсыз әсері ашып көрсетелген. Мұндай қорғау жүйесінің бағдарламалық жасақтамаға енгізуі пайдалылығы көрсетірілген.

Зерттеу, ең алдымен, халықтың сауаттылығын көтеруге арналған, сондай-ақ болашақ қорғау жүйесің даму әсерін арттыруға бағытталған және өнімнің авторлық құқықты қорғау үшін балама жолдары ұсынылған.

Annotation

In this thesis work were considered expired copyright protected (DRM-protection). Studied the working principle, the methods of hacking and counter onomu, identified strengths and weaknesses, and also discloses a negative impact on the protection of the bona fide users, and the system as a whole. the feasibility of such a system of protection was also considered in the software.

The study is primarily aimed at improving the literacy of the population, as well as the impact on the future development of security systems and offer alternative ways to protect the copyright of the product.

Содержание

Введение

Развитие науки и техники породило немало проблем, связанных с авторским правом. Появление первых записывающих медиаустройств, а вскоре и первых оптических носителей дало резкий толчок в развитии так называемого “пиратского” продукта.

Прогресс не стоит на месте и сегодня мы имеем такое понятие как Интернет, которое уже давно вошло в нашу жизнь.

Интернет олицетворяют неким компьютерным вором, для которого нет ни каких преград, и готового присвоить всю "информационную собственность", призывая принять самые жесткие правовые меры по защите от него. Но также Интернет создал идеальный информационный мир, в котором нет границ и непонимания между жителями этого виртуального пространства, и государству рекомендуется не вмешиваться и позволить ему существовать по своим особым законам.

В настоящее время Интернет переполнен материалами, небезупречными с точки зрения авторского права. Как правило, целью Интернета является предоставление информации для других пользователей. Интернет уникален тем, что в любое время дня и ночи можно неоднократно знакомиться с материалами, размещенными в Сети, копировать их, отсылать кому-нибудь при условии, что у вас есть выход в Интернет. И в этом кроется проблема авторского права в сети Интернет. В Сети находится множество контента размещенных без согласия их авторов.

Для защиты интеллектуальной собственности в это неспокойное время пришли так называемые технические средства защиты авторских прав, в мире же известной как DRM.

DRM была призвана защитить авторское право на медиаконтент, но с момента своего появления и до наших дней она не только не справляется со своей задачей, но и приносит вред пользователям и системе в целом.

В странах запада идет ожесточенная борьба с этой проблемой, и эта борьба приносит плоды, многие крупные игроки рынка уже начинают отказываться от DRM.

Статистика показывает, что уровень осведомленности о таком явлении, как DRM среди населения Казахстана очень низок и данная работа призвана решить эту проблему. А также выявить все негативные факторы DRM-защиты, и на основе этих данных предоставить правообладателям альтернативные методы борьбы с цифровым пиратством.

1. Интеллектуальная собственность в интернете

1.1 Понятие авторского права

Авторское право является составляющей частью права интеллектуальной собственности. Авторское право содержит в себе само авторское право, а также смежные права.

Закон Республики Казахстан гласит, что «авторское право на произведение науки, литературы и искусства возникает в силу факта его создания. Для возникновения и осуществления авторского права не требуется регистрации произведения, иного специального оформления произведения или соблюдения каких-либо формальностей» [1]. Но все же, чтобы доказать авторство и иметь право защиты интеллектуальной собственности, нужно произвести регистрацию интеллектуальной собственности, осуществляемые государственными и негосударственными учреждениями.

Согласно пункту 1 статьи 6 Закона Республики Казахстан от 10 июня 1996 № 6-1 «Об авторском праве и смежных правах», авторское право распространяется на произведения науки, литературы и искусства, а также на программы для ЭВМ, которые являются результатом творческой деятельности, независимо от назначения и достоинства произведения, а также от способа его выражения. [1]

Пункт 2 статьи 6 Закона N 6-1 определяет, что авторское право может распространяться, как и на опубликованные, так и на неопубликованные произведения, существующие в какой-либо объективной форме. [1]

При создании и использовании произведений у авторов возникает серьезный вопрос об обеспечении защиты собственных авторских прав. Предмет управления имущественными авторскими правами лежит в создании и деятельности авторских обществ.

Авторские права — это совокупность прав автора (или правообладателя), которые закреплены функционирующим законодательством и направленны на использование произведения, а также на реализацию личных неимущественных прав автора.

В настоящее время, согласно статье 978 Гражданского Кодекса Республики Казахстан, к авторским правам приравниваются исключительные права на использование произведения в любой форме и любым способом.

При использовании произведения автор имеет право разрешить и/или запретить третьим лицам воспроизводить, публично показывать/исполнять, переводить произведения, а также осуществлять иные, не противоречащие законам действия.

Становление авторских прав в первую очередь связано с созданием самого произведения. Права автора – это совокупность имущественных и личных неимущественных прав (Согласно Закону Республики Казахстан «Об

авторском праве и смежных правах»). Если же произведение было создано не одним автором, право принадлежит каждому из авторов. [1]

1.2 Особенности реализации авторских прав в Интернете

Бурное развитие Интернета многократно расширяет информационные возможности не только личности, но и общества. В данный момент Интернет – это главный источник информации в мире.

Литература, музыка, фильмы, программное обеспечение, базы данных, компьютерные игры, все это имеется в Интернете. И это приносит огромную пользу обществу, так как можно оперативно и с минимальными затратами ознакомиться с этой информацией, что способствует развитию личности. Но чаще всего, при распространении различных объектов в Интернете, возникает острый вопрос о нарушении авторских прав, и становится ясно, что свобода информации может приносить не только пользу, но и вред.

Среди пользователей всемирной сети есть те, кто пользуется Интернетом только для поиска и получения необходимых сведений и услуг, а также те, кто не только впитывает информацию, но и сам создает её.

И это порождает проблему, проблему авторского права. Хоть и произведено множество исследований этой проблемы, ограниченно решить этот вопрос сегодня практически невозможно. По мнению одних, Интернет должен следовать обычным законам, другие же говорят, что авторские права в Интернете – вещь виртуальная, и тратить силы на их доказательство не стоит, да и доказать их не получится. На данный момент отсутствуют законодательные нормы для регулирования деятельности в мировой информационной паутине. Поэтому в Интернете чаще всего нарушаются права на объекты интеллектуальной собственности, в частности, авторские права физических и юридических лиц.

В некоторых национальных законах об авторском праве существуют положение: любое копирование продукта недопустимо без ведома правообладателя. Понятно, что загрузка информации из Интернета не может происходить с ведома правообладателя. Это и составляет сложность создания правовых норм регулирования деятельности в Интернете, ведь необходимо сочетать свободу доступа к информации и её безопасность.

При помаши ЭВМ создаются документы, которые могут служить формой авторских произведений. Сам Интернет основан на компьютерных технологиях, которые могут выступать, как и средства ведения коммерческой деятельности, так и являться объектом этой деятельности. Электронная коммерция также поднимает вопрос о защите авторских прав при проведении маркетинговой политики в сети, использовании авторских разработок при создании сайтов и баннеров, борьбе со спамом и т.д.

Интернет активно формирует мировое информационное пространство. С каждым годом увеличивается действие крупных информационных конгломератов, объединяющие системы создания и распространения информации. Интернет концентрирует практически всю информацию по всем

областям человеческой жизнедеятельности: деловая, образовательная, развлекательная, газеты и журналы, базы данных, электронная почта, электронные библиотеки, информационные ресурсы государственных и негосударственных учреждений.

Отлечительной чертой регулирования правовых отношений в Интернете в перво-наперво определяется особенностью создания и распространения информации в электронном виде. Физически свойств носителей информации могут меняться в виртуальной среде (отображение информации онлайн не требует наличия твердого носителя) и, из этого следуют новые юридические особенности и свойства информации как объекта правовых взаимоотношений. Сам Интернете не может иметь свои права и нести обязанности, так как Интернет не является ни зарегистрированной организацией, ни юридическим лицом. Но тем не менее, информация в Интернете в силу права принадлежит различным субъектам.

Сегодня информация, получаемая посредством Интернета, начинает охватываться законодательством об авторском праве. Такую информацию составляют новости, программное обеспечение, произведение литературы, графического искусства и даже электронная почта. Пользователь, не учитывающий эти обстоятельства, сталкивается с серьезными проблемами.

Множество произведений не только впервые публикуются в Интернете, но и распространяются только в нем. Данная сеть все чаще и чаще становится конфликтной средой для авторов и правообладателей.

Пользователи, разработчики сайтов и провайдеры, распространяющие информацию в сети, даже если они добросовестны, могут не знать о требованиях авторского законодательства и нередко нарушать его.

Кроме классических проблем реализации и защиты авторского права (например, проблемы плагиата) Интернет порождает совершенно новые задачи. Одни связаны с международным аспектом, ведь у Интернета практически нет границ, другие – с уточнением и даже реформированием самого авторского законодательства под влиянием все время изменяющихся представлений о некоторых аспектах авторского права.

Дизайн и содержание web-страницы, в том числе: ссылки, оригинальный текст, графика, аудио, видео, HTML и другие языки, списки, составленные организацией или отдельным гражданином, и другие уникальные элементы, все это является объектами авторских прав.

На одной единственной странице может находиться множество объектов авторских прав. И при этом для каждого копирования необходимо будет получить согласие у обладателей этих прав, что практически невозможно.

Поэтому если вы хотите скопировать информацию с сайта, прежде всего нужно обратить внимание на сообщение об авторском праве, которое обычно размещается на самой странице. В этом сообщении четко описано, можете ли вы свободно пользоваться материалом, то есть копировать и вставлять его в другие документы, загружать материал из сети, распечатывать его и насколько широко это можно сделать.

Если же сообщение об авторском праве отсутствует или же операция, которую вы хотите произвести, не предусмотрена сообщением, то вам необходимо получить специальное разрешение. Для этого можно воспользоваться электронной почтой и выслать запрос на адрес разработчика интересующего вас сайта. Получить такое разрешение несложно, особенно если материал является частью рекламной страницы. Однако существуют и отказы в разрешении, если речь идет об информации, касающейся собственности. Британские специалисты утверждают, что запрашивание подобных разрешений должно практиковаться даже в школах.

На некоторых сайтах используются специальные механизмы контроля операций, производимыми пользователями. Например, многие сайты дают возможность распечатать саму страницы, но не дают возможность выполнять такие команды, как «вырезать», «вставить», «сохранить как».

Как только начался просмотр web-страницы, то тут же происходит и копирование информации. Копируется же она в оперативную память компьютера, во буферы браузера и сети. Если у вас есть доступ на определенную защищенную (например, паролем) страницу, тогда можно говорить, что эти копии разрешены. В то же время нет никаких четких оснований считать эти копии законными, ведь не существуют четкие законные основания по этому поводу.

Распечатка страницы также требует предварительного разрешения со стороны правообладателей. Однако данное действие уже может быть разрешено, — стоит лишь просмотреть сообщение об авторском праве, расположенной на данной Интернет-странице.

Следующим важным этапом защиты авторского права в сети является возможность создания выхода с одной Интернет-страницы на другую, то есть создание гипертекстовых ссылок. Создавая гипертекстовую ссылку, вы не создаете копии в прямом смысле этого слова, но можете позволить кому-либо другому сделать копию с материала, к которому вы подсоединены. Язык HTML дает возможность добавить изображение с другого сайта к собственному таким образом, что оно будет частью вашей страницы. Копированием такие действия назвать нельзя, но де-факто такая операция походит на кражу.

Закона о гипертекстовых ссылках пока нет. В такой ситуации нужно руководствоваться принципами вежливости и порядочности. Прежде чем подсоединиться к другому сайту, следует предупредить об этом его правообладателя.

Создавая страницу в Интернете, вы не имеете права на:

- публикацию содержания чужих сайтов;
- копирование и размещение информации из различных ресурсов для создания собственного;
- копирование и объединение информации из различных ресурсов для создания собственного;
- включение в свой документ чужих электронные материалов;

- изменение или редактирования чужой цифровой корреспонденции со сменой смысла;

- копирование и включение чужих списков на свою страницу;

- копирование и включение различной графической информации с других сайтов на свою страницу без соответствующего разрешения.

Таким образом, множество аспектов проблемы авторского права в сети Интернет ждут своего окончательного решения, а возрастающая интеграция информационного сервиса и Интернета требует повышенного внимания к таким аспектам авторского права, как электронное копирование информации, ее распечатывание, создание гипертекстовых ссылок.

Сегодня происходит пересмотр подхода к правовому регулированию Интернета, обсуждаются вопросы о новом разделе законодательства — информационном праве, которое закрепит правовую охрану информации в Интернете.

При активном участии государства, ряд проблем можно начинать решать уже сегодня. Вне всякого сомнения, Интернет относится к сфере, лежащей за пределами юрисдикции государства. А случаи принятия каких-либо ограничений воспринимаются мировой общественностью как покушение на естественную свободу обмена информацией в сети Интернет. Но все же, любое государство не лишено возможности вводить правила и устанавливать ограничения. в отношениях с собственными гражданами и организациями.

В нынешних реалиях Интернета наиболее эффективной является не правовая, а программная защита. И многие юристы подтверждают это.

Техническими средствами защиты авторских прав признаются технические устройства или их компоненты, а также программное обеспечение, которые могут контролировать доступ к произведению, предотвращать или ограничивать выполнение действий, которые не разрешены автором или иным правообладателем в отношении продукта.

В отношении продукта авторского права не допускается:

1) выполнение без разрешения автора или иного правообладателя действий, связанных с устранением ограничения использования продукта, которые были установлены путем применения технических средств защиты авторских прав;

2) изготовление, распространение, предоставление во временное безвозмездное пользование, импорт, реклама любой технологии, любого технического устройства или их компонентов, использование таких технических средств в целях получения прибыли либо оказание соответствующих услуг, если в результате таких действий становится невозможным использование технических средств защиты авторских прав либо эти технические средства не смогут обеспечить надлежащую защиту указанных прав.

В конечном итоге данные проблемы привели к постановке нового вопроса, вопроса о праве «виртуального пространства». Подразумевается, что речь идет не о чем-то несуществующем, вымышленном (виртуальном), а о

вполне себе реальном праве, отношения которых должны быть комплексно отрегулированы. Время создания такой системы норм еще не наступило, но фактически она уже складывается. И это не удивительного, потому что появление новых правовых институтов и правовых структур непосредственно предназначено для общества соответствующих связей и увеличением достаточного обслуживающего правового массива. Но все же, говорить о самостоятельности этой отрасли права пока рано.

В результате все приходит к одному единственному выводу – сам по себе Интернет как компьютерная сеть не является каким-либо новым объектом права, который можно было бы поставить в один ряд, например, с регулированием исключительных прав, права собственности или деликатной ответственности. Не может быть Интернет в строгом смысле и объектом гражданского права подобно имуществу, информации или правам на результаты интеллектуальной деятельности. Но в будущем, это не исключает возможности появления некоторых факторов социальной жизни, которые благодаря развитию Интернета потребуют специфической регламентации в рамках отдельного отраслевого (или более частного) регулирования. Подобно тому, как сто - двести лет назад выявилась социально значимая проблема, связанная с охраной прав авторов литературных произведений, что к настоящему времени привело к созданию целого нормативного массива, относящегося к так называемым исключительным правам "интеллектуальной собственности". К сожалению, пока еще преждевременно предполагать, что именно может потребовать столь принципиального изменения точки зрения на Интернет в целом как на возможный объект права.

2. DRM как средство защиты авторских прав

С нелегальным копированием и распространением предмета авторского права можно бороться различными способами. В настоящее время широкое распространение получили технические средства защиты авторских прав, также известной как DRM, что расшифровывается как Digital Rights Management (Управление Цифровыми Правами)

Согласно статье 2 главы 1 закона Республики Казахстан от 10 июня 1996 года № 6-I «Об авторском праве и смежных правах» техническое средство защиты авторского права – это техническое (программно-техническое) устройство или его компоненты, контролирующие доступ к произведениям или объектам смежных прав, предотвращающие либо ограничивающие осуществление действий, которые не разрешены автором, обладателем смежных прав или иным обладателем исключительных прав в отношении произведений или объектов смежных прав;

Хорошая DRM-защита доставляет большие проблемы пиратам и, в итоге, приводит владельцев авторских прав к требуемой цели (получению прибыли). Но проблемы достаются не только пиратам, но и законопослушным гражданам и системе в целом.

На рисунке 1 приведены графики получения прибыли от продаж незащищённого и защищённого продуктов. Как видно из графиков, если продукт плохо защищён, то он достаточно быстро поддается взлому, и на рынке появляется нелегальная копия продукта, которая не позволяет лицензионной завоевать рынок, и продажи легального продукта снижаются. Если же продукт защищён хорошо, то взлом занимает длительное время, что позволяет продукту достичь требуемого уровня продаж и удерживаться на рынке.

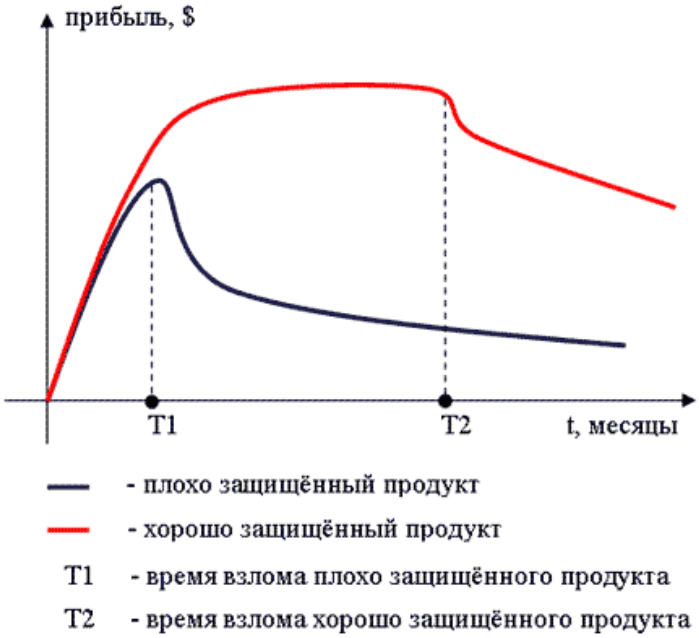

Рисунок 1 – Динамическая зависимость прибыли от степени защищённости продукта

Технологии защиты не стоят на месте. Если же первые версии защиты ломалась в течении нескольких дней, то сегодня время взлома новой защиты считается месяцами.

Рассмотрим основные требования, предъявляемые к современной DRM-защите:

- Высокий запас прочности. Известно, что абсолютно надежной защиты не существует, но выбранная система должна обеспечить достаточную фору во времени до своего взлома.

- Защита без привязки к аппаратной конфигурации ПК, поскольку персональный компьютер не есть вещь в себе, и его отдельные компоненты могут и должны быть заменяемыми по мере старения;

- Защита без использования дорогих дополнительных аппаратных приспособлений, которые повышают стоимость защиты, а стало быть, и конечного продукта;

- Должна быть основана на оригинальных принципах защиты от взлома. Показателем критерия служит факт не взлома данной защиты, либо взлома, но не всеми возможными способами;

- Не препятствовать свободному копированию защищенных данных (должна запрещать только несанкционированный запуск и распространение);

- Защита должна быть подобрана с учетом особенностей рынка ПО.

- Стоимость защиты для разработчика и покупателя должна соотноситься со стоимостью программы.

2.1 Виды и характеристики современных средств защиты авторских прав

2.1.1 Программные и аппаратные методы защиты

Различают программные и аппаратные методы защиты. К программным относятся методы, в которых не затрагиваются физические характеристики носителей информации, специальное оборудование и т.п. К аппаратным же относятся методы с использованием специального оборудования (USB-донглы) или физические особенности информационных носителей (компакт-диски), для идентификации оригинальной версии программы и защиты продукта от нелегального копирования и распространения.

2.1.2 Устойчивость к прямому копированию

Основа любой защиты – это ее способность распознавания физического или цифрового носителя, с которым она попала к пользователю. То есть, способность отличить данный носитель от пиратской копии. Уровень защиты на данном этапе должен быть достаточно высоким, для того чтобы характеристика носителя, не воспроизводилась, то есть не была сэмулирована любыми средствами побитового копирования. Эффективностью данного этапа защиты служит стойкость защиты к элементарному копированию, когда пользователю достаточно запустить эмулятор компакт-дисков, и не о чем больше не думать.

Многие системы защиты используют различные физические метки, по которым программа распознает валидность оптического носителя. Физические метки плохо копируются, но специальные драйвера неплохо их эмулируют, что показывает, что защита, которая основана только на физических метках, очень уязвима. Еще, как правило, все системы защиты представляют собой "вставную челюсть". То есть модуль, который отвечает за идентификацию диска, в большинстве случаев не может противостоять хакерской атаке, в то время как менее важные участки кода оказываются хорошо защищенными.

Более эффективным считается способ, при котором на оптический носитель не наносятся метки, то есть, оптический носитель может спокойно копироваться и распространяться, но доступ к информации можно получить только с наличием оригинального диска.

Данный вид зашиты основывается, что любой оптический накопитель (CD/DVD/R/RW) имеет ряд уникальных характеристик, свойственных только одному диску, и при копировании на другой диск эти характеристики естественно теряются.

2.1.3 Устойчивость к взлому

Если защита не поддается копированию и эмулированию, хакер прибегает к использованию механизмов дизассемблирования и отладки, для проведения анализа приложения и выделения логики её защиты, а после нейтрализует её. Соответственно, если защита устойчива к прямому копированию, то есть у неё имеется надежный способ работы с уникальными характеристиками носителя, это позволяет ей однозначно идентифицировать достоверность представленного носителя. Но если же модуль распознавания не защищен как надо, то она не выдержит даже любительской хакерской атаки, и рассматривать такую систему защиты не стоит.

2.1.4 Аппаратные ключи

Часто для реализации защиты от несанкционированного распространения используются электронные ключи (HASP) или Sentinel.

HASP – это программно-аппартный комплекс, содержащий в себе код, процедуры или любые другие уникальные данные, благодаря которым защита распознает легальность запуска.

Современные электронные ключи подключаются практически ко всем портам компьютера, но чаще всего используются USB-ключи

Основой ключей HASP служит специальная микросхема (микроконтроллер) - ASIC (Application Specific Integrated Circuit), содержащая уникальный для каждого ключа алгоритм работы.

Принцип защиты состоит в том, что в процессе запуска защищённая программа опрашивает подключённый к компьютеру ключ HASP. Если HASP возвращает правильный ответ и работает по требуемому алгоритму, выполнение программы продолжается. Если же неправильный, она может завершаться, переходить в демо-режим или закрывать доступ к каким-либо функциям программы.

Значительная часть моделей ключей HASP содержат энергонезависимую программно-перезаписываемую память (EEPROM). В зависимости от реализации HASP размер памяти может составлять 1-4 килобита.

Энергонезависимой память, которая содержится в HASP, дает возможность её программирования размещая внутри модуля различные функции и процедуры, или хранить дополнительные ключи, а также:

- управлять доступом к различным программным модулям и пакетам программ;
- выдавать уникальный номер каждому пользователю защищенных программ;
- сдавать программы в аренду и распространять их демо-версии с ограничением количества запусков;
- хранить в ключе пароли, фрагменты кода программы, значения переменных и другую важную информацию.

У каждого ключа HASP с памятью имеется уникальный идентификатор (ID-number), доступный для считывания защищёнными программами. Идентификаторы дают возможность различать пользователей программы. Проверяя в программе идентификатор HASP, пользователь может предпринимать те или иные действия в зависимости от наличия конкретного ключа. Идентификатор присваивается электронному ключу в процессе изготовления, что делает невозможным его замену, но гарантирует надежную защиту от повтора. С использованием идентификатора можно шифровать содержимое памяти и использовать возможность ее удаленного перепрограммирования.

Система HASP позволяет защищать программное обеспечение двумя различными способами: автоматически (стандартно) и вручную (через специальный API).

2.1.5 Предоставление SDK со стороны разработчиков защиты

SDK (Software Developer Kit, комплект разработчика ПО) позволяет более детально познакомиться с внедряемой DRM-защитой. Для этого SDK включает полную техническую документацию, описание утилит и средства разработки.

В комплект разработчика входит все, что нужно для внедрения представляемой системы защиты в собственный программный продукт. Это – детальные примеры, фрагменты кода, поддержка различных операционных систем и средств разработки.

Также SDK может включать и аппаратные устройства (hardware) для построения тестовых проектов. Примером может служить SDK HASP, который содержит в себе демонстрационные ключи.

SDK дает возможность внедрения DRM на начальных стадиях разработки продукта. Защиту, которая была встроена в приложение на последнем этапе разработки, не составляет особого труда взломать. Обойти же защиту, встроенную в само приложение, ой как не просто.

Что же дает разработчикам использование SDK? SDK может применяться для защиты отдельных участков кода или данных приложения, а также может устанавливать места проверки легальности запуска (например, обращение к ключу/диску).

SDK тоже бывают разные. При выборе защиты с предоставлением SDK следует рассчитать стоимость комплекта, разузнать о наличие поддержки, и, одно из важных, - простоту изучения. Последний критерий является спорным, ведь эффективность SDK прямо пропорциональна сложности изучения.

2.1.6 Лицензирование

Суть любой из систем лицензирования заключается в том, что после установки программного обеспечения на локальный компьютер пользователю необходимо получить от производителя ключ, который был бы тем или иным образом привязан к компьютеру (в основном), хотя это может быть и банальный пароль, и вовсе необязательно этот ключ привязан к компьютеру.

Системы лицензирования могут быть как "одна лицензия - один компьютер" так и "одна лицензия - один пользователь". Как правило, для этих целей используют механизм заполнения анкеты на сайте производителя (для идентификации пользователей) + пересылку (на тот же сайт) специального идентификатора компьютера, на основе которого и выполняется генерация ключа. Как правило, в ключе, в зашифрованном виде, содержится информация о пользователе, продукте, числе лицензий и другой информации.

Одним из ярких представителей системы лицензирования является Globertrotter FlexLM. Система лицензирования многих компаний, работающих на корпоративном рынке.

FlexLM (и ряд других компаний) предлагает два типа лицензий: Floating и NodeLocked.

Floating – плавающий тип лицензий. Данный вид лицензий устанавливается на сервер и оговаривает число одновременно работающих устройств в сети. То есть, имея 10 лицензий, продукт может использоваться одновременно только на 10 устройствах, но установлен он может быть на неограниченное число рабочих мест. Данный тип защиты позволяет продукту распространяться, но запускать его одновременно можно только на ограниченном числе устройств. Защита наиболее эффективна в корпоративном сегменте, где работает много специалистов. Налицо экономическая выгода, ведь не все пользователи одновременно работают с одними и теми же программами. В случае корпоративного применения экономия может составлять 50%-80% от общего числа необходимых лицензий.

NodeLocked – фиксированный тип лицензий. Данный способ защиты позволяет работать только на одном устройстве. Способ хорошо подходит для индивидуальных пользователей, которые работают только с одним устройством.

В силу вышеуказанных причин, стоимость Floating лицензии выше, чем NodeLocked. Также лицензии делятся на две категории: постоянные и временные.

Временные – не привязываются к устройству, не ограничивают функциональность программного продукта, однако ограничивают время пробной эксплуатации.

Постоянные – снимают все временные ограничения, но привязываются к конкретному устройству.

Модули лицензирования типа FlexLM являются встроенными системами, то есть они встраиваются в программный продукт уже после его окончательной разработки. Лицензия FlexLM запрашивается либо в момент старта продукта, либо при выполнении определенных операций. В момент запуска проверяется достоверность ключа и характеристики компьютера, если он привязан к таковым.

Для идентификации компьютера используется множество способов. Вот основные способы привязок:

- к серийному номеру жесткого диска;

- к MAC-адресу сетевой карты;
- к контрольной сумме BIOS;
- к различным характеристикам системы.

Метод авторизации через Интернет

В силу того, что IT отрасль переходит от платных продуктов к продуктам Shareware, то есть условно-бесплатным продуктам, интересным представляется метод получения ключей через Интернет.

Многие компании позволяют пользователям работать со своими продуктами некоторое время бесплатно. Это обоснованное решение, поскольку стоимость лицензий может быть достаточно высокой, а пользователь, зачастую не знает, нужен ему продукт или нет.

FlexLM имеет в своем арсенале временные ключи, не ограничивающие функционал продукта, но имеющий один существенный недостаток. FlexLM можно обойти простой сменой дат, то есть пользователь может подкрутить системное время назад, с целью продления срока эксплуатации. Соответственно, в чистом виде подобная защита не подходит большинству компаний-разработчиков ПО.

В отличие от классического временного ключа, активация через Интернет имеет ряд неоспоримых преимуществ. Схема работы продукта выглядит следующим образом:

При первоначальном запуске, пользователя просят зарегистрироваться, заполнив анкету. После заполнения через Интернет происходит активация продукта для данного устройства. Далее сервер начинает отслеживать либо число запусков продукта, либо ведет обратный отсчет времени эксплуатации. Учитывается, что при каждом запуске пользовательский компьютер автоматически через Интернет посылает запрос на сервер регистрации о возможности запуска. Если ключи еще валидны, сервер дает "добро" на запуск приложения.

Данный способ позволяет получить список пользователей, работающих с временными версиями, ограничить сроки эксплуатации и подвигнуть пользователей к официальному приобретению продукта.

К техническим достоинствам можно отнести то, что ни переустановка продукта, ни чистка реестра не позволяют повторно использовать пробный период для одной и той же машины. Недостаткам защиты же является требование выхода в Интернет на момент запуска защищенного продукта. В случае отсутствия соединения продукт не запускается.

2.2 Способы взлома и противодействия

2.2.1 Побитовое копирование CD

Данный способ даже вряд ли можно назвать взломом. Пользователь (не всегда злоумышленник) пытается скопировать имеющийся у него носитель (компакт-диск) с целью создания копии для личного пользования или для распространения.

Для осуществления этого могут использоваться различные программы, зачастую входящие в поставку устройств CD-R/RW. Это и официальные Easy CD Creator и Nero, и полуофициальные (полухакерские) CloneCD и BlindRead.

Защита должна уметь противодействовать данному способу, так как с него обычно и начинается взлом, поскольку программ-копировщиков, способных скопировать диски с примитивной защитой, великое множество.

Существуют два способа противодействия копированию. Первый заключается в том, что на диск записывается определенная метка, которая не копируется обычными средствами (например, создается нестабильный сегмент, который не читается носителем, а раз не читается, то и скопированным быть также не может). К сожалению, данный способ не всегда надёжен, поскольку уже существуют программы "продвинутого" копирования (те же CloneCD и BlindRead), которые способны пропускать подобные места (замещать нестабильные области произвольными данными) и проводить копирование до конца.

Второй способ основывается на том, что ничего никуда записывать не надо, а надо лишь определенным образом запоминать физические характеристики диска, которые просто невозможно воспроизвести любым копированием, точнее диск сам по себе копируется, но уже с другой физической структурой. Соответственно, пользователь может спокойно клонировать диски, но ключевым будет тот, который был официально куплен. То есть, в данном случае, диск будет использоваться как ключ доступа к информации. Для примера реализации данного метода можно упомянуть продукт "Большая Советская Энциклопедия", состоящий из 3 дисков, информация с которых свободно копируется, но работа с энциклопедией возможна только при наличии первого диска.

2.2.2 Эмулирование CD

Данный подход заключается в формировании виртуальных драйверов устройств и имитации обращения к диску. Это уже чистой воды взлом, поскольку для нормальной работы вскрытого приложения в систему инсталлируется специальный драйвер, который имитирует обращение к нестабильной метке на диске и возвращает вскрытой программе именно те данные, которые она ожидает "увидеть". Подобный способ довольно часто применяется на первых порах, когда хакеру известен способ получения метки на диске, но ему не очень хочется разбираться с программой методом дизассемблирования.

Противодействием может служить работа с устройствами чтения/записи на низком уровне, когда невозможно перехватить вызовы к оборудованию. Здесь нужно еще внести одно пояснение: для того, чтобы защищенному приложению обратиться к CD, и проверить его на наличие специальной метки, необходимо воспользоваться одной из функций чтения/записи, которые предоставляет Windows. Хакерами уже наработан ряд механизмов, позволяющих перехватывать стандартные обращения к функциям Windows, а

раз можно перехватить обращение, значит можно имитировать чтение, целиком заменяя стандартные вызовы на собственные. Как говорилось выше, противодействием данному способу взлома может быть только обращение к накопителю не через стандартные вызовы.

2.2.3 Эмулирование электронных ключей (HASP)

Эмулирование аппаратных ключей происходит так же, как и эмулирование оптического носителя, но основная сложность заключается в эмулировании процесса обмена ключ - драйвер - ключ. Одной из важных возможностей электронных ключей защиты является кодирование данных с помощью ключа, используемых защищенным приложением.

Противодействием эмуляции является частое использование функций кодирования данных, применение декодированных данных непосредственно в работе защищенного приложения (без предварительного сравнения).

В случае аппаратной реализации алгоритмов кодирования полная эмуляция ключей защиты становится практически невозможна, злоумышленники стараются снять защиту путем взлома программного модуля.

Эмулирование данного типа устройств осуществляется так же, как и для CD, если не проще. Вызовы к электронному ключу проще (относительно) перехватить и построить эмулятор.

Противодействием является программирование доступа к ключу на низком уровне, без использования стандартных механизмов, но и тут необходимо быть осторожными.

2.2.4 Взлом программного модуля

Если не удается скопировать приложение (а также сэмулировать диск/HASP), а способ защиты приложения неизвестен, то хакер переходит на следующий уровень взлома, исследует логику самой программы, с целью, анализа всего кода приложения, чтобы выделить блок защиты и деактивировать его.

Взлом программ выполняют двумя основными способами: отладка (исполнение по шагам) и дизассемблирование.

Отладка – это специальный режим, который создается специальным программным обеспечением, называемым отладчиком, который позволяет выполнить любое приложение пошагово, передавая ему всю среду и делать вид, что приложение работает только с системой, а сам отладчик невидим. Не только хакеры используют отладку, но и разработчики тоже, ведь отладка является единственным способом выявить неправильную работу программы. Используя эту благую идею, хакеры проводят анализ кода приложения для поиска модуля защиты.

Это так называемый, пошаговый режим исполнения, или, другими словами интерактивный. Второй же способ – дизассемблирование - это преобразования исполняемых модулей в .asm код, понятный человеку. В этом

случае хакер получает распечатку того, что делает приложение. Обычно этот код очень громоздкий, из-за чего распечатка может быть очень и очень длинной, но никто и не говорит, что ломание защиты – это легко.

Хакеры активно используют эти два механизма взлома, потому что иногда приложение проще воспроизвести по шагам, а иногда проще получить листинг и произвести его анализ.

2.2.5 Отладка

Видов отладчиков очень много: от отладчиков, которые являются частью среды разработки, до отладчиков-эмуляторов, которые полностью помещают отлаживаемое приложение в аналитическую среду, что дает разработчику (или хакеру) полный анализ работы приложения. С другой же стороны, подобный отладчик настолько четко эмулирует среду, что приложение, отлаживаемое под ним, считает, что работает напрямую с системой (типичный пример подобного отладчика - NuMega SoftIce).

Способов противодействия отладке также немало. Основная задача этих способов – сделать работу отладчика либо совсем невозможной, либо максимально её усложнить. Опишем основные способы противодействия:

Замусоривание кода программы. Способ, когда в код программы вносятся специальные сложные функции и вызовы, создающие вид работы с накопителями, но по факту ничего не делающие. Способ в первую очередь направлен на отвлечение злоумышленника от кода самой системы защиты.

Использование многопоточности. Еще один эффективный способ защиты, который использует возможности Windows по параллельному исполнению функций. Любое приложение может выполняться линейно, инструкция за инструкцией, и легко отлаживаться, а может разбиваться на несколько потоков, которые будут исполняться одновременно, что существенно будет усложнять отладку и работу хакера. А если потоки будут еще и шифроваться, то хакер надолго завязнет, пытаясь вскрыть приложение.

Подавление изменения операционной среды. При выполнении, программа сама несколько раз перенастраивает среду окружения, или вообще отказывается работать в измененной среде. Так как не все отладчики способны на 100% эмулировать среду системы, и если отлаживаемое приложение будет перенастраивать настройки среды, то рано или поздно "неправильный" отладчик может дать сбой.

Изменение определенных регистров процессора, на которые отладчики не могут реагировать. Также, как и с предыдущим способом. Отладчик – это программа, и также как все программы пользуется и операционной системой, и процессором, который один на всех. Так, если изменять определенные регистры процессора, с которыми отладчик не сможет работать, то можно существенно "подпортить" его здоровье.

Противодействие постановке контрольных точек. Специальный механизм микропроцессора, при помощи которого можно исследовать не всю программу сначала, а, например, только её определенный кусок. Для этого в нужном месте программы ставят специальный вызов (контрольную точку -

Breakpoint), который передает управление отладчику. Недостаток способа кроется в том, что для осуществления прерывания в код исследуемого приложения надо внести изменение. А если приложение время от времени проверяет себя на наличие контрольных точек, то сделать подобное будет весьма и весьма непросто.

2.2.6 Дизассемблеры и дамперы

Про дизассемблер было сказано выше, а дампер – это практически тот же самый дизассемблер, только транслирует он не файл, находящийся на диске в Ассемблерный код, а содержимое оперативной памяти в момент нормального исполнения приложения (то есть, пройдены все защиты). При дамппе хакер не бороться с механизмами, противодействующими отладке, он всего лишь ждет, когда приложение пройдет все проверки на легальность запуска, проверяя метки на диске, и начинает нормально функционировать. В этот момент дампер и снимает "чистенький" код без примесей. К всеобщей радости не все защиты могут просто так себя раскрыть! Об этом поподробнее:

Шифрование. Самый простой и эффективный способ противодействия. Подразумевает, что определенная часть кода никогда не появляется в свободном виде. Код дешифруется только перед передачей ему управления. То есть вся программа или ее часть находится в зашифрованном виде, а расшифровывается только перед тем как исполниться. Соответственно, чтобы проанализировать ее код надо воспользоваться отладчиком, а его работу можно очень и очень осложнить (см. выше)!

Шифрование и дешифрование (динамическое изменение кода). Более продвинутый способ шифрования, который не просто дешифрует часть кода при исполнении, но и шифрует его обратно, как только он был исполнен. При такой защите хакеру придется проводить все время с отладчиком, и взлом защиты затянется на очень и очень долгое время.

Использование виртуальных машин. Еще одна модернизация шифрования. Способ заключается в том, чтобы не просто шифровать и дешифровать целые фрагменты кода целиком, а делать это покомандно, подобно тому, как действует отладчик или виртуальная машина: взять код, преобразовать в машинный и передать на исполнение, и так пока весь модуль не будет исполнен. Этот способ гораздо эффективнее предыдущих, так как функции приложения вообще никогда не бывают открытыми для хакера. Естественно, что его трудно реализовать, но реализовав, можно оградить себя от посягательств любых хакеров. В этом способе кроется также и недостаток - снижение производительности, ведь на подобное транслирование требуется много времени, и, соответственно, способ хорош для защиты только критических участков кода.

Дополнительные способы противодействия

Здесь уже даются общие вводные, ведь защита может быть эффективной только тогда, когда каждый ее модуль написан на совесть с использованием

различных ухищрений. То есть все рецепты, о которых говорилось выше, должны в той или иной форме присутствовать в любой системе.

Использовать для хранения данных защиты системные ресурсы Windows: дополнительную память, выделяемую для параметров окон и локальные хранилища потоков. Суть способа состоит в нестандартном использовании стандартных областей, скажем, хранить ключи, пароли… и т.п., совсем не там, где их будут искать при взломе в первую очередь.

Использовать операции сравнения нестандартными способами, во избежание их явного присутствия. Для сравнения есть определенные инструкции микропроцессора, о которых знают и разработчики и хакеры. А если попытаться использовать нестандартные виды сравнения, то можно слегка запутать хакера, ожидающего стандартного ответа.

Избегать обращений к переменным, относящимся к защите напрямую. То есть использовать любые косвенные способы доступа к специальным областям.

Использовать метод "зеркалирования" событий, то есть применять нестандартные действия на стандартные вызовы. Об этом говорилось выше.

Использовать для шифрования надежные, проверенные временем алгоритмы.

2.3 Обзор современных средств защиты

AsProtect

Разработчиком данного протектора является Алексей Солодовников. Разработана в среде Borland Delphi 7. В коммерческих продуктах не применялась. Поддается антиотладке, ломается одним битом в условном переходе. Оригинальная точка входа нарушена. Стартовый код “размазан” по аллочной памяти, вместе с мусором. Взлом – X-code inject на MOV BYTE PTR DS:[EBX],0E9 (в районе 1512C6Fh), регистр EBX – где вставить опкод JMP’a, регистр ECX – его операнд, там единственный раз ссылка на PUSH EBP(OEP), из секции .text на бывшем месте OEP JMP в кучу на PUSH EBP, сигнатура нужной кучи в начале- “0F B7 FF 80”. Дамп на месте. PUSH EBP в аллочной памяти. Восстановление start code.

Особенности. Эмуляция основных системных API (чаще всего копируются первые 5 байт в выделенную аллочную область, и через PUSH71XXXXXX, RET управление передается на след. Инструкцию в API)

Точка входа:
PUSH [address]
CALL [address with one RET]
RET
Т.Е. ЭКВИВАЛЕНТНО:
JMP [address]

SafeDisc

Разработчик этой DRM MacrovisionCorp. Разработана в среде Microsoft Visual C++6.0. Применялась в основном в играх издательства EA/DICE: Battlefield 2 и другие. Поддается антиотладке: IsDP, RDTSC. Оригинальная точка входа не нарушена. Дамп нерабочий: INT3 на некоторых инструкциях (в основном связанных с выделением аллоков). Взлом – дисассемблировать SEH – обработчик и найти таблицу эмулирования. Оригинальная точка входа не нарушена.

Особенности. Элементы межпроцессного взаимодействия. Эмуляция украденных инструкций через INT3

На месте точки входа ставиться переход на расшифрованную область памяти (при отсутствии отладчика)

SecuROM

Разработчик – Sony DADC. Разработана в среде Microsoft Visual C++6.0. Применялась во многих компьютерных играх начиная с начала 2000-х годов. Среди них: GTA III, GTA IV, Command & Conquer 3: Tiberium Wars, Bioshock и другие. Поддается антиотладке: хук IsDP, обращение к отладочным регистрам, взвод флага трассировки. Оригинальная точка входа не нарушена. Дамп нерабочий: часть внутренних вызовов, часть вызовов API идут через VM, которая извлекает указатель (ESP+8) на момент входа и дополняет недостающие элементы в памяти, совершает правку операндов прыжков на требуемую процедуру, после выполнения передает управление на требуемую процедуру.

Особенности. VM в аллочной области. Перестроенная частично секция кода защищаемой программы. Часть процедур программы перенесена в секцию .securom.

PUSHFD и код, проверяющий наличие программных точек остановок.

StarForce

Разработчик – Protection Technology. Разработана в среде Microsoft Visual C++6.0. Применялась в играх российских и европейских разработчиков: STALKER, SplinterCell и другие. Антиотладка возможна: IsDP, защита от аппаратных и программных точек останова. Взлом: X-code inject в protect.dll, патчить драйвера. CRC и побайтовая проверка некоторых частей protect.dll(шкурка). Взлом: править байт перехода (найти можно по cmp …, 4D5Ah и cmp …,5045h – проверяет хидер), переименовать оригинальный protect.dll в P.D, например, затем в нашей protect.dll через GetModuleFileName возвратить имя P.D, тем самым заставив защиту проверять целостность переименованного файла.

Все функции защиты заложены в protect.dll (или dll по названию приложения), драйвера служат для обнаружения SoftICE, обращения к аппаратным точкам останова (DR0-DR4), патчингу ядра и драйверов приводов, выкидывания BSOD

Точка входа – переход в protect.dll.

TagesSA

Разработана в ThompsonCorparation в среде Microsoft Visual C++7.0. Защита была применена в игре Witcher. Не поддается анти-дебаггингу. Оригинальная точка входа не нарушена. Дамп рабочий.

Оссобенности.

Основные функции проверки в hc.dll, выделение аллока, копирование замусоренного кода. Взлом: поднимитесь на самую верхнюю инструкцию hc.dll и поставьте точку останова. Дальше без остановок. Для остановки: первой управление получает и держит hc.dll (00087330h), ловится через ntdll (процедура перехода в DllMain,WinMain)

3. Взлом SecuROM. Теория и практика

Для закрепления проведенного исследования произведем взлом одной из популярнейших систем защиты SecuROM 7.3 на примере игры Command & Conquer 3: Tiberium Wars (с патчем 1.9).

3.1 Словарь терминов

Для дальнейшей работы нужно ознакомиться со следующими терминами.

X-код – это осмысленная совокупность байт, внедренных посторонним лицом или программой в целевой код процесса, для выполнения определенной задачи.

Аллочная память (heap – куча) – область памяти, выделенная по требованию через API VirtualAlloc

Хук (hook) – грубо говоря, элемент взаимодействия между процессами, при котором один процесс может внедрить свой код в адресное пространство другого процесса. Понятия X-code и hook схожи.

SecuROM 7 VM – совокупность кода, состоящая из “островков” и использующая “хранилища” для восстановления требуемых при обращении к нему данных, таких как адреса вызываемых внутренних функций защищаемой программы, данные самой вызываемой функции, запросы к WinAPI и сокрытие некоторых операций в специальных островках. Прямое назначение – анти-дампинг защищаемого приложения.

“Хранилища” – структурная единица, специально отведенные SecuROM’ом участки памяти, в которых храниться вся информация для работы VM. Обращение к ресурсу хранилища идет как смещение от начала данного хранилища. “Главное хранилище” – основной юнит, на котором построена работа виртуальной машины.

На рисунке ниже – Хранилище №1.5, которое всегда является поставщиком RVA всех островков и выходов из VM.

“Ресурс в хранилище” – специальный адрес, имеющий свое строго определенное смещение от начала хранилища. Это определение я отношу по существу к главному хранилищу с диапазоном смещений от 0 до 2Ch.

Ячейка	Зашифр offset	Декод offset	Адрес	Контрольный б...
13D803F8	F8540003	FE150000	151B0000	000000FE
13D803F4	FFE80003	FFFA0000	17000000	000000FD
13D803F0	F4940003	FD250000	142B0000	000000FC
13D803EC	F4D00003	FD340000	143A0000	000000FB
13D803E8	AC904D43	EB241350	022A1350	000000FA
13D803E4	F5E00003	FD780000	147E0000	000000F9
13D803E0	FF780003	FFDE0000	16E40000	000000F8
13D803DC	AC907603	EB241D80	022A1D80	000000F7
13D803D8	F9280003	FE4A0000	15500000	000000F6
13D803D4	AC9034C3	EB240D30	022A0D30	000000F5
13D803D0	FBFC0003	FEFF0000	16050000	000000F4
13D803CC	F7AC0003	FDEB0000	14F10000	000000F3
13D803C8	F8040003	FE010000	15070000	000000F2
13D803C4	AC905563	EB241558	022A1558	000000F1

Рисунок 2 – Хранилище №1.5

“Островки” – структурная единица, обособленный участок кода в VM, конец которого отождествляется JMP EAX(EDI), RET или непосредственным указателем, иногда после прыжков стоят обращения к реверсерам или мусорный код, впрочем, читать первые необязательно. Их единый стандарт: работа с ресурсами (+4) и (+10) в “главном хранилище” и “хранилищем №1.5” для получения адреса следующего “островка”. Выполняемые функции одного островка могут быть следующими:

1) Копирование адресов(чисел) из ячейки в ячейку главного хранилища или копирование в ячейки из запрашиваемых адресов вне хранилищ.

2) Декодирование заложенных зашифрованных адресов в виртуальном стеке

3) Перезапись адресов в стеке

4) Правка стрелки

“База запроса” – формирует собственный запрос к VM из двух аргументов.

1) Указатель на Виртуальный стек (обвертка)

2) Адрес, куда требуется передать управление после выхода из VM.

“Виртуальный стек” – концепция аналогично обычному машинному стеку. Отведенная область памяти и размерность в одно двойное слово (DWORD). Но важным различием является перевернутая структура виртуального: его верхушка – всегда младший адрес, и от младшего к старшим адресам смещается виртуальный указатель, Т.е. секуромовский виртуальный стек как раз и “НЕ растет сверху вниз”, а “уменьшается снизу-вверх”.

В своем виртуальном стеке для VM существует только два типа данных:

1) *Контрольное слово.* Содержание байт может варьироваться (зависит от “загруженности” островка) от 1 до 3 для каждого действия (1 байт – как стандарт). Прежде всего, это следующее значение для ROL-байта на следующем островке. Далее по значимости идет смещение, по которому из Хранилища №1.5 извлекается адрес следующего островка или выхода из VM. Следующим по значимости идут смещения для ячеек в главном хранилище –

туда заносятся/извлекаются адреса или какие-либо промежуточные значения в работе VM. Все что касается контрольных байт-смещений:

После извлечения контрольного байта над ним будет произведено декодирование истинного его значения, естественно с помощью текущего значения ROL-байта(т.е. в виртуальном стеке они хранятся зашифрованными если можно так выразится), затем будет произведено увеличение истинного значения в 4 раза(x4) - в подавляющем большинстве случаев эту операция выполняет асм инструкция SHL *REG_32*, 2 (побитовый сдвиг влево на 2 позиции равен целочисленному умножению на 4!), полученное сырое смещение будет сложено с начальным адресом нужного хранилища.

2) *Зашифрованный адрес/число*. Одновременное использование всех двух типов данных островком говорит о том, что он выполняется процедуру расшифровки адреса (в этом случае в контрольном слове подготовлен байт, который укажет на ячейку в главном хранилище куда оно будет положено, ну и повторюсь что для остальных островков коим требуется этот адрес в их контрольных словах также будет указатель на ту же самую ячейку).

С помощью револьверного алгоритма (в EDX – взятый из виртуального стека *Зашифрованный адрес/число; CL- ROL-байт)*:

```
ROL DL, CL
ROL EDX, 8
```

Который в сумме выполнится 4 раза, последним XOR с определенным ключом выполнится расшифровка.

Справедливы следующие правила:

- На один островок – одно контрольное слово. На один островок с алгоритмом расшифровки – один зашифрованный адрес в виртуальном стеке.

- Для островка с алгоритмом расшифровки в контрольном слове всегда предусмотрен байт-указатель на свободную ячейку в главном хранилище, куда требуется поместить результат. Естественно островки, которые будут пользоваться этим результатом, уведомлены через свои контрольные слова из какого смещения требуется прочесть адрес (естественно из-за накручивания ROL-байта значения контрольных байт-указателей в этом случае не совпадают, к тому же место такого контрольного байта-указателя в контрольном слове всегда может быть различным)

- В случае использования островком только контрольного слова указатель виртуального стека (он же ресурс (+4) в главном хранилище) съезжает вниз к старшим адресам на 4 байта (одно двойное слово).

- В случае использования островком контрольного слова и зашифрованного адреса указатель виртуального стека съезжает вниз к старшим адресам на 8 байт (два двойных слова). По инструкциям ADD DWORD PTR DS:[(+4)], 8 и ADD DWORD PTR DS:[(+4)], 4 в обфусцированном коде островка очень просто различить его назначение!

“Обвертка на виртуальный стек” – адрес, который всегда заносится в 1й аргумент базы запросов. По нему первый островок определяет начало виртуального стека.

“Виртуальный указатель” или *V-ESP* – он же ресурс (+4) в главном хранилище. Аналогично прямому назначению регистра ESP указывает в виртуальном стеке его верхушку.

Рисунок 3 – Виртуальный указатель

Именно по островок сначала и узнает, какое двойное слово ему предназначено.

Правила для хранилищ:

*Максимально допустимое смещение для Хранилища №1.5 и Главного хранилища не может превышать 3FCh, т.к. FFh*4 = 3FCh*

(FFh(255) – максимально допустимое значение в одном байте - контрольном байте)

Минимально допустимое смещение для Главного хранилища не может быть менее 2Ch (минимальные контрольный байт для ячеек = Bh), т.к. ниже область занята под специальную структуру ресурсов самого Главного Хранилища.

“Прыжок в середину инструкции” – известный трюк, основанный на правиле дисассемблирования двоичного кода. По аналогии с примером, который Крис приводил в своей известной книге “Искусство дисассемблирования” (Изд-во: БВХ-Петербург,2008): у нас есть некоторый набор букв – ПОДОРОГЕЕХАЛАМАШИНА. Процессор начинает перебирать варианты для того, чтобы построить из него нормальное предложение (правильно дисассемблировать все четыре инструкции):

П ОДОРОГЕЕХАЛАМАШИНА, ПО ДОРОГЕЕХАЛАМАШИНА,… ПО ДОРОГЕ ЕХАЛАМАШИНА… пока не дойдет до единственно правильного варианта - ПО ДОРОГЕ ЕХАЛА МАШИНА. Естественно такой расклад увидит у себя в окне хакер. Заметьте, что слово МАШИНА содержит еще одно знакомое слово – ШИНА. Используя переход в “середину” слова (асм инструкции) МАШИНА, после слова ЕХАЛА, можно построить новое предложение: ПО ДОРОГЕ ЕХАЛА ШИНА.

“Дельта”, “вычисление дельты” – Виртуальная машина всегда располагается по новым адресам (результаты работы VirutalAlloc), естественно нельзя выполнит привязку к конкретным значениям, поэтому возникает

необходимость знать текущее реальное местоположение кода в памяти. Дельта – разность между текущим реальным положением кода и его заявленным положением при компиляции (или от какой либо фиксированного адреса).

Типичный набор инструкций для вычисления дельты: CALL delta
delta: POP EAX
SUB EAX, offset delta //в EAX будет значение дельты

В виртуальной машине Дельта-смещения можно встретить в Хранилище №1.5. Вычисление происходит вычитанием RVA островков от RVA островка входа в VM.

3.2 Подготовительные работы

Все что связано с внедрением X-кода, чаще относят к области вирусописания, поэтому это сложная и объемная тема для разговора. Однако, в нашем случае все намного проще: X-код, который внедряем мы, выполняет одну несложную задачу - показ адреса возврата. Непосредственно внедрять свой X-код будем через отладчик OllyDbg v2 в народе известной как Оля (Ольга, Олька).

Для удобства разобьем нашу большую задачу на отдельные подзадачи.

Задача 1: Получить доступ на запись в секцию кода kernel32

Описание: По стандарту секция кода (.text) имеет атрибуты Read/Execute (чтение/выполнение).

Поэтому код типа, где 77E41C00 – адрес в секции кода kernel32:

MOV BYTE PTR DS: [77E41C00], E9

Вполне на законном основании вызовет исключение access_violation, что не входит в наши планы. Так как у меня две винды, я решил патчить вистовский kernel32 в 2k3 оффлайн (он же bithack). Вариант с использованием WinAPI VirtualProtect все-таки более предпочтителен (в исходниках X-кода он есть!), но я считаю, что на руку нестандартные решения. В 2k3 открытая на запись секция кода kernel32 приводит к краху некоторые приложения (VisualStudio 2008, OllyDbg 1.10), зато данную шалость в висте они спокойно воспринимают! Жалко, что Windows не для хакеров пишут – закрытые на запись секции кода и непропатченный IsDebuggerPresent не есть хорошо :)

Решение: Заходим в 2k3. Запускаем PeTools. Не забываем выставить свои права доступа к библиотеке. Добавляем к секции кода Write атрибут и пересчитываем контрольную сумму. Аналогом PeTools может послужить утилита editbin, если установлен VisualStudio: например, для пересчета checksum наберите в командной строке “editbin /RELEASE E:\Windows\system32\kernel32.dll”. Если виста запустилась и в раскладке карты памяти в ольге, в столбце Access секция кода kernel32 любого процесса помечена как RWE, стало быть, все сделано правильно!

Задача 2: Найти свободное место для X-кода в cnc3game.dat; найти адрес для установки перехода, который передаст управление на наш X-код; установить, какие байты будут перекрыты переходом; установить, по какому адресу потребуется возвратить управление после установки перехода

Описание: В cnc3game.dat 11 секций (PE Header не при делах, хотя при большом желании внедриться можно и туда). Первые 6 однозначно не подходят (text – зашифрованный код самой игрушки; data, rdata, rsrc – область данных программы; tls – TLS Callback; rts.ver – имеет атрибут только на чтение). Ars – при беглом просмотре код, подвержненный обфускации, все атрибуты доступа, оставим в покое. Est – здесь находиться точка входа, все атрибуты доступа, кандидатура подходит. Адресат внедрения – чаще всего, цепочка нулей в конце, оставленных для выравнивания секций и никем не используемых. Artem – название походит на шутку разработчиков, небольшой и не имеющий осмысленности код, секция имеет все атрибуты доступа, но лучше ее не трогать. Celare – однозначно все данные и ресурсы SecuROM'а, пропускаем. Одноименная секция SecuROM по утверждению Оли содержит таблицу импорта, имеет все атрибуты доступа, не трогаем. Теперь передача управления. Как показывает практика, переход лучше всего делать с распаковщика. В отличие от остальных структур, его код не модифицируется в процессе выполнения (хотя в нашем мире все непостоянно). Можно просто переставить точку входа на наш X-код, но для нас в этом нет необходимости.

Решение: Хвост секции est всегда свободен. В моем случае, начиная с адреса 11DB6D0h, будет располагаться наш X-код. Ставим точку останова при доступе к секции text (распаковщик ее обязательно затронет и даст знать о себе). Запускаем приложение. Оказываемся в функции с адресом, в моем случае, равным 11DB1F9h. Я выбрал адрес 11DB27Eh. Переход будет длинным, значит один байт E9 плюс 32х битный операнд, который займет 4 байта, значит всего 5 байт. По адресу 11DB27Eh идут ADD EDI,ESI и ADD ESI,EBX которые занимают 2 байта и MOVZX EBX,BYTE PTR DS:[ECX+7] которая занимает целых 4 байта. Наш переход полностью перекрывает первые две и последним байтом задевает MOVZX, значит, эмулировать придется все эти три команды, запоминаем их. Теперь вычислим, куда потребуется возвратить управление. В сумме все три, будучи эмулируемые нами, команды занимают 2+2+4 = 8 байт. Значит, 11DB27Eh + 8h = 11DB286h. К этому адресу мы и возвратимся. Пишем по адресу 11DB27E: JMP 11DB6D0 и добавляем три NOP. Стоит отметить, что никто нам не мешает возвратиться сразу на адрес после прыжка (точнее на первый NOP), впрочем, большого смысла в этом нет.

Задача 3: Первая часть X-кода (подготовительная). Проэмулировать перекрытые переходом JMP 11DB6D0 инструкции. Передать управление собственно на X-код. Вычислить адрес GetDriveTypeA. Установить какие байты будут перекрыты переходом, ведущим ко второй части нашего X-кода и проэмулировать их. Организовать вычисление операнда инструкции JMP для 2го перехода. Корректно записать инструкцию прыжка на 2ю часть X-кода. Возвратить управление.

Описание/Решение: Три аспекта объясню подробно. В нашем случае идеально первая часть X-кода должна выполниться один раз, больше и не нужно. Но распаковщик работает в цикле, о чем говорит регистр EAX, который с каждым разом инкрементируется. Мы сначала проверим регистр EAX на одно

фиксированное значение (в моем случае 6Ch), если равно, то передаем управление X-коду. Получение адреса функции из библиотеки это две API:

GetModuleHandle и GetProcAdress

В регистре EAX – адрес требуемой API.

С перекрытыми байтами GetDriveTypeA и вообще в большей части Windows API есть полезная особенность компилятора VC++! Первые три инструкции в экспортируемых функциях системных библиотек винды это MOV EDI, EDI; PUSH EBP; MOV EBP,ESP которые и дадут в сумме нужные нам 5 байт, и предпринимать каких либо дополнительных мер не потребуется! Разбираем операнд инструкции JMP. Вся проблема в том, что kernel32, равно как и другие системные библиотеки может “плясать” по адресному пространству процесса даже в пределах одной винды. Отсюда выходит, что пропатчить верхушку GetDriveTypeA фиксированным значением нельзя, ведь, в конце концов, на месте адреса, который сейчас ты видишь в отладчике, вполне реально окажется, не то, что хотелось, к тому же и операнд прыжка укажет в совсем другое место! Почему? Напоминаю, что в качестве операнда для любой из инструкций прыжка указывается количество байт, которые надо перепрыгнуть, а не адрес назначения! Дополнительно: x86 операнд записывается в обратном порядке и независимо от условного или безусловного перехода операнды определяются одинаково! Чтобы правильно вычислить операнд, я сначала посчитал разницу между точкой входа в упомянутой WinAPI и адресом назначения прыжка, затем так как прыжок будет идти в сторону младших адресов, плюс, учитывая размер инструкции JMP LONG (5 байт), получил окончательное значение:

```
SUB EAX,cnc3game.011DB6E3
MOV EDI,-5 // EDI = FFFFFFFBh
SUB EDI,EAX
```

Задача 4: Вторая часть X-кода (основная). Извлечь адрес возврата. Преобразовать байты адреса возврата в ACSII, для корректного вывода. Вывести на экран адрес возврата. Корректно осуществить возврат в системную библиотеку и эмуляцию перекрытых байт.

Описание/Решение: Процесс запустился, первая часть X-кода успешно выполнилась, управление передается на GetDriveTypeA, затем через поставленный нами переход мы оказываемся в начале второй части нашего представления. Адрес 011DB6E3h. Мы разберем самые хардкорные аспекты его действия. Если ты уже понял, то перед нами стоит проблема корректного отображения адреса возврата. Действительно, если в качестве аргумента для текста API MessageBoxA, скормить адрес возврата, то в сообщении нашего сателлита вместо адреса возврата будет откровенная ерунда (под словом “ерунда” понимается ASCIIZ строка, которая начинается с адреса возврата), ведь для винды 00404001 это адрес/операнд, откуда надо считать строку, но не сама строка! Так как я не знаю API, которая могла бы выполнить такое преобразование (_itoa опустим), было принято решение написать самостоятельно код, который мог бы выполнить требуемую операцию. На

самом деле это несложно! Я рассуждал так: каждому символу в ACSII соответствует свой код. Для того чтобы узнать, как будет закодирован каждый байт адреса возврата, потребуется организовать цикл. Коды чисел от 0 до 9 в ACSII имеют значение 30-39h, кодам букв (в нашем случае это числа) от A до F присвоены соответственно значения 41-46h. Руководящая идея – организовать сравнение байт адреса возврата и эталонного значения, посредством инкрементов последнего и его кодов в ACSII таблице. Если эталонный байт = сравниваемому, то мы пишем два значения их кода по специальному отведенному адресу (в байте две цифры, поэтому мы используем старшую и младшую часть регистра ECX для ACSII кода каждой из двух циферок). Последнее замечание касается перескока с 9 на A - в ASCII таблице их коды 39h и 41h соответственно, а при инкременте 39h следующее число 3Ah, за которым закреплен другой символ.

Чтобы не упустить момент, скажу сразу, что кроме MessageBoxA существуют еще множество вариантов с выводом информации:

1) С помощью CreateWindowEx. Создать окно с контролами (Edit,ListBox) и отсылать в ставку с помощью SendMessage секретную информацию.

2) В файл с помощью CreateFile/WriteFile/CloseHandle и их низкоуровневых аналогов. Сюда же можно приписать именованные каналы(PIPE).

3) Весьма оригинальный: с помощью DirectX (Draw или даже 3D). Благо игрушка сама подключает библиотеку. Однако здесь нужно уметь работать с интерфейсом от МелкоСофт, да и удобнее уже будет написать свою отдельную Dll’ку для работы и инжектить ее с самой игрой.

Задача 5: Проверить работу внедренного X-кода.

Правильный ответ: Кроме зеленого логотипа игрушки на заднем плане, первым мы должны увидеть наш сателлит – MessageBox сообщающий нам 8 hex цифр (рисунок 3). Это и есть первый вызов GetDriveTypeA и его точка возврата в обратном порядке (если X-код перед глазами, то надеюсь, понимаешь, почему он транслирует адрес в обратном направлении). Первый адрес не относится к cnc3game.dat и лежит где-то в ntdll. Затем MessageBox выскочит еще n раз, где n – количество логических устройств на твоей машине (вместе с виртуальными приводами, естественно). После выходит окошко от SecuROM’а с просьбой вставить нужный диск. После нажатия “Повтор” сателлит покажется n раз, затем снова окно с просьбой вставить требуемый диск.

Неправильные ответы и их возможные причины: Сателлит не появился, но игрушка работает – Вторая часть X-кода не получила управления -> Первая часть X-кода ошибочно поставила переход в другом месте. Ошибка программы – Необрабатываемое исключение -> Кодовая секция kernel32 без атрибута записи ИЛИ X-код ошибочно передал управление в другую область памяти. Сателлит появился, но вместо 8 циферок откровенная ерунда – вместо ACSII строки адрес возврата. Адрес в сателлите указывает на несуществующую область памяти - адрес возврата декодирован в ACSII строку неверно.

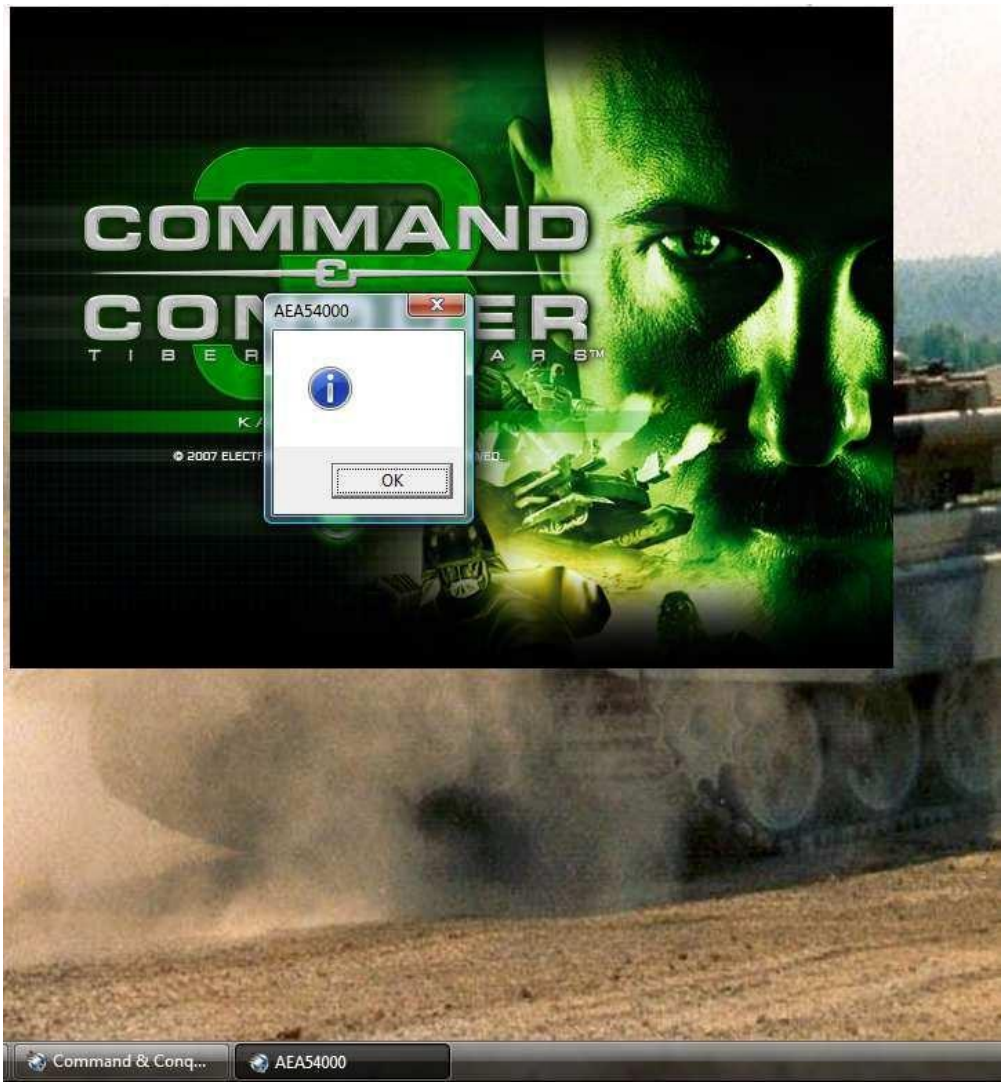

Рисунок 4 – Проверка работы внедрённого X-кода

Наш шпион уже передает нам hex адреса. Осталось корректировать область его работы и записывать результаты на бумагу. Помимо уже поднадоевшей API GetDriveTypeA можно немного переделать код и с нужными параметрами организовать прослушку CreateThread, DriveIoControl, CreateFileA, RegQueryValueExA, KiUserCallBackDispatcher. Одним словом, работаем, как стандартный API шпион. А в чем тогда разница между оным и методом в нашем случае? Типичный API-spy использует стандартные процедуры меж процессного взаимодействия (ReadProcessMemory, WriteProcessMemory или ниже) и может быть легко обнаружен протекторами. Причем хороших API-spy немного, и они хорошо известны разработчикам защитных комплексов (причем этот факт легко проверить – прослушайте CreateFileW в протекторе). С появлением новых версий защитных систем, типичные методы работы шпионажа за API медленно и постепенно уходят в прошлое. Инновационным и самым современным подходом, который лишен всех недостатков обычных API-spy и достигает самой максимальной скрытности, за счет использования ресурсов самого протектора стал X-code injection.

Собственно, что представляет собой SecuROM 7 изначально? Код, подверженный обфускации, причем мусора не очень много, ставка сделана на ассемблерные трюки. Можно сказать, это кладезь всевозможных ухищрений. Например:

```
MOV EAX,11D7B1C // Начальный адрес зоны проверки
{
MOV ESI,DWORD PTR DS:[EAX] //грузим след DWORD
ADD DWORD PTR SS:[ESP+10],ESI //складываем с предыдущим DWORD'ом
ADD EAX,4 // the next offset DWORD
DEC WORD PTR SS:[ESP+0C] // 114 DWORD'ов над сложить
JNE SHORT 011D7B76 // - while (dword [ESP+0Ch] != 0)
}
OR BYTE PTR SS:[ESP+10],01 // добавляем в младший байт единицу
SUB DWORD PTR SS:[ESP+10],933 // вычитаем "контрольную сумму", пасьянс сошелся если: DWORD [ESP+10] <= 0. (Вообще-то там всегда нуль должен быть, отрицательное число – разработчики просто решили подстраховаться)
PUSHFD //сохраним флаги (EFL = 206h)
… //код, предназначенный для отвода глаз
POPFD //вытаскиваем флаги (EFL = 206h)
JBE SHORT 011D7BC1 //так сошелся ли все-таки наш пасьянс? (Zero Flag(Z) = 1 или(и) Carry Flag(C) = 1?)
```

Также, чаще смотрите, что вы трассируете:

```
00C49160   PUSHFD
00C49161   MOV EAX,DWORD PTR SS:[ESP]
00C49164   NOP
00C49165   TEST AH,1

00C49168 JE SHORT cnc3game.00C4916F 00C4916A MOV ECX,7BE
00C4916F   XOR EAX,EAX
```

Оказывается, в SONY DADC еще нашли весьма оригинальную замену инструкции MOV ESI, DWORD PTR DS:[ESI]:

```
MOV DWORD PTR SS:[ESP],ESI
XOR ESI,DWORD PTR DS:[ESI]
XOR ESI,DWORD PTR SS:[ESP]
```

Если поставить break on memory access на IsDebuggerPresent, то потрассировав процедуру, которая попадется в сети, можно увидеть:

```
SUB ESI, EAX
LEA ECX, DWORD PTR DS: [ESI+EBX-5]
MOV BYTE PTR DS: [EAX], 0E9
MOV DWORD PTR DS: [EAX+1], ECX
```

SecuROM 7 активно используют прием с постановкой переходников: вычисляют адрес API, вычисляют длину перехода, прибавляют к нему 5 байт (MOV EDI,EDI; PUSH EBP; MOV EBP, ESP), записывают переходник, ну и

эмулируют открытие кадра стека. Теперь, если протектору требуется вызов API, то он делает это через переходник, эмулируя открытие кадра стека у себя и попадая на 5 байт “раньше” в область библиотеки с требуемой API. Вот мы и ознакомились с основным приемом против API-spy. Дело в том, что последние на инструкцию MOV EDI,EDI привыкли вешать хуки, которая в нашем раскладе никогда не получит управление! Правда по каким-то неизвестным причинам, SecuROM 7 не задействует переходники в своей работе. Но на будущее в своей реализации X-code injection надо учитывать этот трюк. Отличительной особенностью этой версии протектора являются illegal instruction UD2 (плюс аппаратные точки останова с ними), на которые повешены SEH-обработчики. Коих я насчитал всего два (00C996B0h и 00C99702h). Если проанализировать их, то все они ведут к JMP… и выполняют абсолютно одну и ту же работу. Но обо всем по порядку.

Цепочка GetDriveTypeA (набор приводов) -> CreateFileA (открытие привода на секторном уровне) -> DeviceIoControl (проверка на наличие диска в заданном приводе, количество секторов с дорожками) ->DeviceIoControl (первичные 5 заходов) -> MOV DWORD PTR DS:[EBX+EDI*8+4],EAX (Если проэмулировать наличие диска в приводе, находим головную часть проверки)-> QueryPerfomanceCounter(вторая часть проверки) и синхронные потоки…

CMP BYTE PTR SS:[EBP-52],BL // проверка на наличие диска в приводе

Еще пару интересных моментов с GetDriveTypeA и за hardware breakpoints. Зная расположение WinAPI и чтение ветки HKLM\System\MountedDevices в защитном механизме можно скрыть от протектора виртуальный привод, если это потребуется или сразу все приводы!

Второй момент, связан непосредственно с вызовом самой API и ей подобных – в нескольких местах протектор вызывает их через серию JMP и стандартную тройку LoadLirary/GetModuleHandle/GetProcAddress. Что касается аппаратных точек останова, как было сказано выше, на них также повешены упомянутые SEH-обработчики, это ведет к тому, как без ZwContinue и подобных операций с контекстом SecuROM 7 выясняет наличие отладки.

Переходим к нахождению OEP и снятию дампа. Нам понадобиться Daemon Tools (или любой другой эмулятор привода) и мини-дамп оригинального диска игрушки. Хотя, если окончательно доразобрать процедуру проверки и поправить нужные переходы, то можно обойтись без эмуляции. Потребуется любой ценой попасть в оригинальную точку входа и снять корректный дамп, и эта задача действительно решаема. Поможет нетронутый никем cnc3.exe, который сообщает нам, что разработчики писали игрушку на Microsoft Visual C++ 7.0.

Точка входа выглядит как:

004628DA CALL 004784B8

004628DF JMP 004626FA

Реально, точка входа находится по операнду прыжка - 004626FAh. Функция по адресу 004784B8h ничем полезным не занимается. Однако в ней мы находим, что она вызывает несколько API (GetSystemTimeAsFileTime,

GetCurrentProcessId…). Для нас это означает только одно – если поставить нашего “шпиона” на GetSystemTimeAsFileTime и после распаковки стартовый код действительно на своем законном месте… Принимаемся за реализацию. Монтируем мини-дамп. Запускаем игру. Нас интересуют все MessageBox после проверки: 00DDCE77,76B414D4,7C34207B, 0040A5AE…

Присоединяемся к процессу Ольгой 1.10 с дампером и переходим по последнему адресу. Перед нами действительно OEP! Получилось! Переходим по прыжку к адресу 0040A006 и дампим наш процесс.. Теперь самое главное: SecuROM 7 имеет полномочия защищаться от дампинга. Дамп естественно получается изначально нерабочим. Беглый анализ приносит известия, что на некоторые вызовы (к примеру, чуть ниже, по адресу 00A400D) ведут не к дочерним процедурам, через серию JMP DWORD PTR DS:[address] и “базу запросов” попадают в аллочную память к всемогущей виртуальной машине(VM). После ее работы двойное слово по адресу прыжка магическим образом меняется и уже указывает на нужную процедуру. Причем еще до обращения к “базе запросов” она уже была по своему адресу. Глядя на бесконечно длинный ужасный код этого виртуального чуда, складывается впечатление – реально ли вообще отвязать SecuROM от программы? А может лучше постараться приклеить выделенную память протектора к игре в качестве новой секции (подавляющее большинство так и поступает, т.к. это наиболее простой способ, но в плане оптимизации – самый жуткий)?

3.3 Процедура проверки диска в приводе

LEVEL 1. ПРОВЕРКА СИГНАТУРЫ

GetDriveTypeA. Как бы намекает, что действие скоро начнется.

CMP EAX, 5 //DRIVE_CDROM

CreateFileA. Открывает заданный диск (Имя файла в виде: \\.\D), получает хендл.

DeviceIoControl (4 РАЗА). Главную роль здесь отвели DeviceIoControl, с чьей помощью можно все что угодно сделать или получить с устройства. Прежде всего важен ControlCode, который может принимать три значения:

1) (IOCTL_SCSI_PASS_THROUGHT_DIRECT) 4D014h или 4D004h. Первоочередной возможностью является информация о физическом наличии диска в устройстве. Затем идет различная информация о дорожках, секторах и тд. Размер буфера всегда равен 50h(80байт).

CMP BYTE PTR SS:[EBP-52],BL //Вставлен ли диск в лоток?

Первая проверка только для подтверждения наличия диска в приводе. Если все ОК, то начинаем процедуру чтения и составления сигнатуры.

На данном этапе, при выполнении WinAPI DeviceIoControl, обращайте внимание на InBuffer(ESP+8): от начала InBuffer по смещению $+14h расположен указатель на структуру спец. буфера, куда, после выполнения вызова, ложатся служебные данные (например, номер сектора).

А теперь… три разводные проверки, для прохождения первого левела:

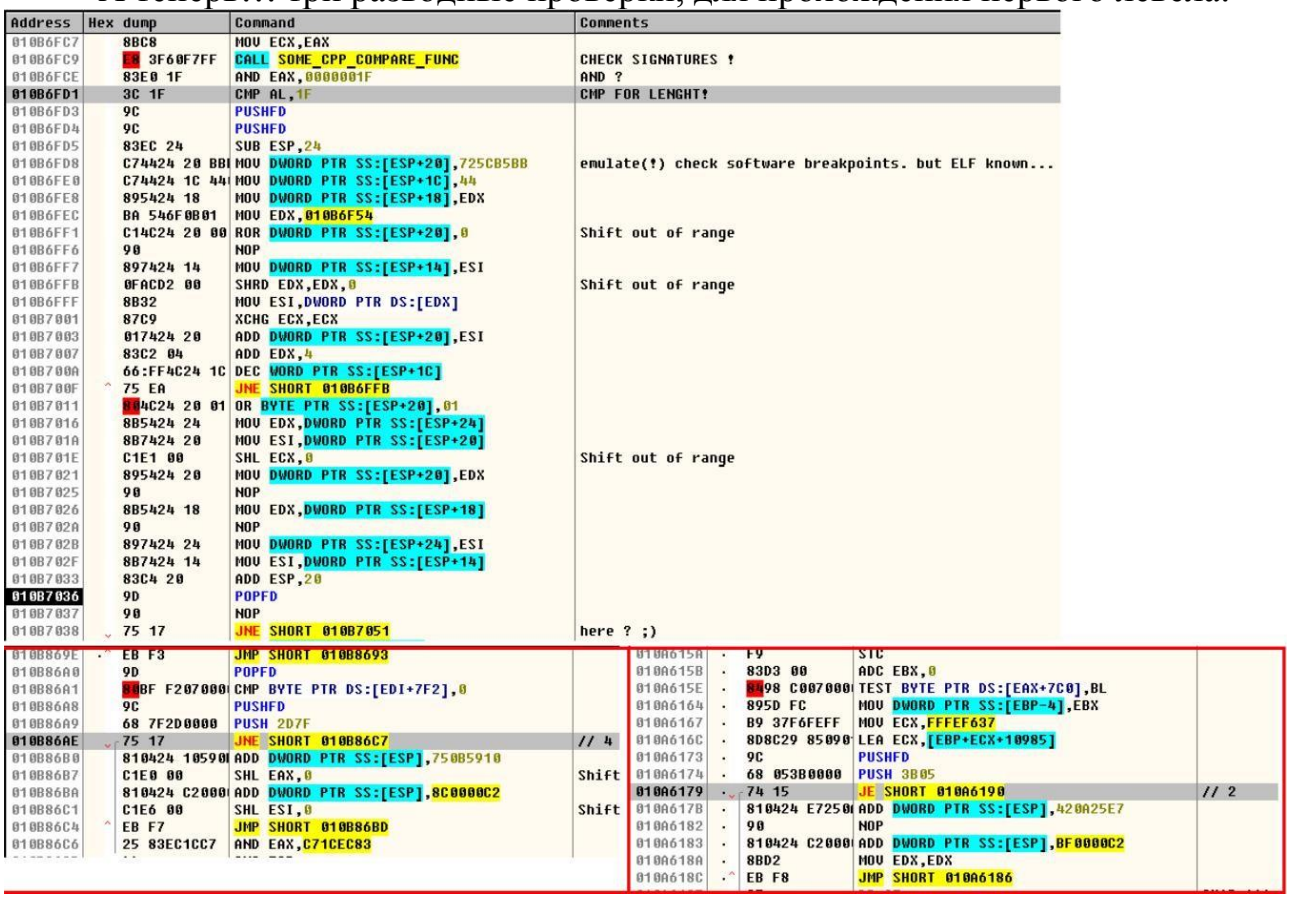

Рисунок 5 – Прохождение первых 3 проверок

Правильное прохождение первой из проверок (и последующих), заставляет SecuROM 7-8 генерировать сообщение с повтором: Cannot authenticate the original disc. Your disc may require a different software version.

Правда, есть еще 4 проверка, но её патчинг не влияет конечный на результат-Level 1 completed!

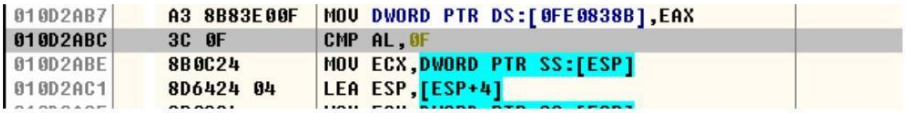

Рисунок 6 – Прохождение 4 проверки

ВАЖНО! Патчить нужно непосредственно операции, влияющие на флаги, а не сами переходы! PUSHFD защита сохраняет состояние EFL для следующих переходов (в 7.33.017 их не более 2х), т.е. условие проверяется несколько раз.

Кстати, характерная деталь в этой версии для первой сигнатурной проверки – «липовый» подсчет контрольной суммы участка в перекрывающем коде.

QueryPerfomanceCounter & QueryPerfomanceFrequency. Начинают считать такты в нормальном (подготовительном режиме).

LEVEL 2. ПРОВЕРКА ГЕОМЕТРИИ(ТАКТОВ) ИЛИ QPC/QPF

SetSystemCursor.

После успешного прохода сигнатурного уровня процесс всегда стал завершаться с кодом выхода 1. На это влияет HKEY_CURRENT_USER\Software\SecuROM. После удаления ветки Key – процесс стал завершаться теперь после прохода геометрической проверки (в Bin таблицы, структуры данных проверок по времени и геометрии, метка диска под XOR и т.д).

Изменяем (010DF7A9) как показано на рисунке ниже (signed int > 0).

```
010DF7A6    897D 64          MOV DWORD PTR SS:[EBP+64],EDI
010DF7A9    803D BF1D2D01 00 CMP BYTE PTR DS:[12D1DBF],0      -- 2fix
010DF7B0    9C               PUSHFD
010DF7B1    68 AF060000      PUSH 6AF
010DF7B6    76 15            JBE SHORT 010DF7CD               not taken
010DF7B8    810424 F70F3300  ADD DWORD PTR SS:[ESP],330FF7
010DF7BF    C1E1 00          SHL ECX,0                        Shift out
010DF7C2    810424 C3E1DA00  ADD DWORD PTR SS:[ESP],00DAE1C3
010DF7C9    EB FA            JMP SHORT 010DF7C5
```

Рисунок 7 – «Фикс» (010DF7A9)

Относительно файлов, вложенных в протектор и создаваемых во временной папке. Доступ во временную папку может быть закрыт, поэтому функции, расположенные в drm_dyndata_7330017.dll, могут быть дублированы в образе.

Ну а сейчас встречаем виртуальную машину! Paul.dll (паша.длл)

3.4 Разбор виртуальной машины

Рисунок 8 – Основное хранилище VM

Ключом, для понимания работы “виртуальной машины” (далее по тексту просто VM) является обычная структура, именуемая как “главное хранилище” (смещение которого от входа в VM чаще всего равно +2000h. В рассматриваемом случае оно равно 20A7000h, вход в VM - 20A5000h соответственно. Это спираль, возле которой танцует основная логика работы

всех “островков” или более объективно – 255 кусков кода, каждый из которых работает по единому шаблону с элементами главного хранилища (+0, +4, +10, 14) и выполняет только одну свою задачу (общее число задач равно четырем, плюс несколько специфических). Очевидно, что островки, хотя по внешнему виду разные, но по факту – дублируются.

Следующие операции на любом островке всегда делаются по умолчанию:

1) Чтение ресурса (+4), операция сложения результата с ресурсом (+C), в итоге получение текущего виртуального указателя ESP на виртуальный стек.

2) Чтение (вытаскивание по аналогии с POP EDX, см. пункт 5) по виртуальному указателю:

А) одного двойного (контрольного) слова, если задача островка не связана с расшифровкой

Б) двух двойных слов. Первое требуется расшифровать, второе – контрольное.

3) Чтение ресурса (+10) - контрольного байта. Если островок действовал по 2-Б, то контрольным байтом будет выполнен первый этап расшифровки, второй этап – маска XOR с одним и тем же ключом. Далее из контрольного слова будет взят, специальный байт (или 3 байта), который будет прибавлен (в дополнение возможно: XOR с определенным ключом, вычитание, сложение) к текущему ROL-байту, новое значение будет сохранено в ресурс (+10) для следующего островка

4) Получение из контрольного слова байта, операция умножения в 4 раза, полученное смещение складывается с адресом хранилища №1.5, оттуда читается двойное слово, над которым будет произведен побитовый сдвиг направо ROR, (операция эквивалентна делению) затем сложение с адресом входа в VM. Так получается адрес следующего островка или выхода из VM. Этот алгоритм стабилен для всех.

5) Как следствие пункта 2, естественно требуется поправить виртуальный указатель ESP, к ресурсу (+4) прибавляется:

А) 4 байта, если было использовано (вытащено) одно двойное слово

Б) 8 байт, если было использовано (вытащено) два двойных слова

В) Исключение: 10(1) байт. Связано с некоторым различием работы VM в SEH-обработчиках самой защиты и VM непосредственно в “тибериумной драке”. Чтобы запутать взломщиков, в первом случае изначально виртуальный стек специально делается “кривым”(не кратен 4) и примерно с десяток “островков” не несут в себе полезной нагрузки. После их выполнения все станет на свои места.

По инструкциям ADD DWORD PTR DS:[(+4)], 8 и ADD DWORD PTR DS:[(+4)],4 легко идентифицировать тип островка!

6) Собственно прыжок на следующий островок JMP EAX(JMP EDI/RET) или выход из VM.

Декодирование полученного смещения из Хранилища №1.5.Типичная операция в конце островка – переход на следующий:

MOV EAX, 1 //EAX = 1 для инструкции CPUID говорит о том, что инструкция возвратит так называемую сигнатуру CPU - информацию о процессоре (модель, степпинг)

CPUID //кодирование/декодирование смещений в таблице Хранилища №1.5 осуществляется при помощи этой асм инструкции!

AND EAX,FFFFFFDF //получаем байт для сдвига через CPUID

ROR EDI, CL ; EDI = C003FFF7/ F0 = FFFDF000 //ДЕКОДИРУЕМ

ADD EDI, DWORD PTR DS:[(+C)] ; EDI = FFFDF000 + 020A5000 = 02084000

/*ПОЛУЧЕННОЕ СМЕЩЕНИЕ СКЛАДЫВАЕМ С АДРЕСОМ ВХОДА В VM*/

JMP EDI ; EDI = 02084000 //ПРЫГАЕМ НА СЛЕДУЮЩИЙ ОСТРОВОК

Для того чтобы положить свой адрес в №1.5 требуется выполнить обратные действия:

1. Вычислить дельту (RVA входа в VM минус RVA начала твоего кода

2.Выполнить:

MOV EAX, 1

CPUID

AND EAX,FFFFFFDF

ROL EDI, CL

Сигнатура ЦП (EAX = 1)

Возвращаемые в регистре EAX биты после выполнения CPUID.

3:0 – Stepping (степпинг процессора) 7:4 – Model (Модель)

11:8 – Family (Семья)

13:12 – Processor Type (тип процессора) 19:16 – Extended Model (расширенная модель) 27:20 - Extended Family (расширенная семья)

У AMD и Intel существуют различные нюансы в значениях, которые возвращают биты.

Другие значения EAX для выполнения CPUID.

EAX = 0(ID поставщика ЦП)

В регистрах EBX, EDX, ECX – первые 12 байт от ASCII строки, которая идентифицирует фирму изготовителя. Например, AuthenticAMD говорит о том, что поставщиком процессора является компания Advanced Micro Devices(AMD).

EAX = 2(Кэш и информация о TLB дескрипторе)

TLB – Translation Lookaside Buffer. В двух словах, этот буфер является КЭШем ЦП для увеличения скорости трансляции виртуальных адресов.

EAX = 3(Серийный номер ЦП)

В зависимости от модели и производителя процессора в EDX:ECX (или EBX:EAX) возвращается серийный номер.

Виртуальный стек SecuROM 7 VM хранит в себе два типа данных:

1) Контрольное слово (Control DWORD). Манипулирует логикой работы VM.

Имеет в себе 4 байта из которых:

- Следующее значение ROL-байта для следующего островка. Всегда присутствует. Как стандарт 1 байт. Исключение: 2 байта (из них “клеится” один байт, который аналогично будет прибавлен к ресурсу (+10)).

- Адрес следующего островка по Хранилищу №1.5 [1 байт] Всегда присутствует. 1 байт. По существу, является закодированным смещением.

- Адрес ячейки в главном хранилище. В зависимости от функции текущего островка. По существу, является закодированным смещением.

2) Зашифрованные данные в размере одного DWORD. Такие как: Адрес запрашиваемой функции, адрес ASCIIZ строки, число. Этот тип используют непосредственно островки, занимающиеся их расшифровкой (с помощью ROL-байта).

Сама VM имеет три режима работы - по количеству выходов из нее.

Способ №1 – когда требуется вызвать внутреннюю функцию (к слову, это может быть и одна асм команда) в самой защищаемой программе;

Способ №1A – практически тоже самое, что и первый, однако его обращение к VM характерно прямым вызовом через регистр EAX (CALL EAX), который у себя формирует базу запроса и сразу переходит в VM (т.е. CALL 00D44F40), после конечного RET управление передается в запрашиваемую WinAPI.

Способ №2 – радикально отличается от первых двух, результаты его работы в регистрах EAX и ECX. База запросов. Самый главный первый аргумент, который является простой “обверткой” и указывает на начало области в секции .secuROM, с которой будет работать VM. Для понимания представьте ее в качестве “виртуального стека”, а ресурс (+4) в “главном хранилище” виртуальным аналогом регистра ESP, который в рассматриваемом примере первоначально устанавливается равным адресу 0155741C. Второй аргумент является адресом возврата, на который передаст управление последний RET в VM, для первых двух способов его заданное значение не используется (адрес 0040365A попадает в середину команды) и будет исправлено VM в стеке на адрес конечной запрашиваемой функций, обратная ситуация для способа №2, адрес возврата будет чуть ниже базы запросов и конечный RET в виртуальной машине передаст управление на него. Из базы запросов работает переход на адрес 00D44F40, своеобразный блокпост VM, который всегда делает прыжок на начало VM. Даже не смотря на то, что существуют его адреса-аналоги, все базы запроса указывают на него и все через него проходят в VM. Начало VM, разбираем суть работы. Первый островок (его я специально оставил как есть, с остальных обфускация снята.). Первым делом обфускация, суть ее проста: математическими операциями получать смещения в главном хранилище (как стандарт). Например:

```
MOV EDX,60
SHR EDX,3 ; EDX = 0Ch //ресурс (+C)
MOV ECX,72
XOR ECX,00000062; ECX = 10h //ресурс (+10)
```

Впрочем, такая простота присуща не всем островкам, на некоторых есть длинный запутанный алгоритм по получению смещения +4. После получения смещений и выполнения операций в главном хранилище, SecuROM возвращает указатель на его начало, вычитая использованное смещение. Стоит отметить, что чаще всего в обфускацию вовлечены два регистра EDX и ECX, предварительно их значения сохраняются в стек.

Собственно, переходим к разбору функционала. Стандартные операции сохранения регистров и флагов, взвод spin-блокировки, говорит о том, что с VM имеет право работать только один поток, остальные будут ожидать сброса блокировки, которая произойдет на выходе из VM. Теперь требуется инициализировать главное хранилище, предварительно обнулив его содержимое. Обратите внимание, что наш виртуальный ESP или ресурс (+4) будет хранить указатель на виртуальный стек в неявном виде, об этом говорит команда по смещению ACh от входа VM, которая вычтет из адреса-указателя адрес входа в VM (точнее говоря, хранится его смещение). Соответственно, чтобы прочитать контрольное слово по виртуальному ESP, машинке требуется обратно прибавить адрес входа в VM, который будет также помещен в главное хранилище в ресурсе (+C). В дополнение к (+4), по смещению +18 лежит “обвертка” виртуального стека, по которой всегда можно установить его начало. После ресурса (+4), еще больше внимания заслуживает ресурс (+10) и непосредственно DWORDs в виртуальном стеке.

По первому, можно увидеть, что первоначально его значение всегда равно 95h. Почему 95h? На самом деле неважно сколько, важно соблюдение правил всей системы. Представьте, что у вас есть два очень простых уравнения:

X+Y = 5 (1) 4+Y = 7 (2)

Если не учитывать, второе, то число 5 можно получить разными способами, однако, правильным будет только один вариант, когда Y = 3, а X = 2. Последний это и есть аналог 95h. Очевидно, что по названию ресурс(+10) связан с асм командой ROL, которая знакома программистам как побитовый сдвиг налево. Она и несет основную функцию декодирования, а с помощью контрольных байт, каждый раз ее второй операнд получает нужный сдвиг. ROL-байт принимает участие в Хранилище №1.5 только для декодирования его реального контрольного байта, а в свою очередь, алгоритм для шифрования/расшифровки смещений, где используется результат работы CPUID в регистре AL с тандемом операции побитового сдвига, но уже направо – ROR, идет отдельно. По поводу остальных ресурсов: так как островки разбросаны по всей выделенной памяти, адрес которой – результат API VirutalAlloc, то существует проблема “что где лежит”. Ее решает ресурс(+0) со только что упомянутым хранилищем №1.5, в котором хранятся неизменные смещения от начала входа в VM всех островков (по сути: вычисленные Дельта-смещения). В нашем случае он равен 01FD1000, и как положено, в главном хранилище хранится его смещение. Не менее важным является и то, что для каждого процессора результат работы CPUID может быть разным, поэтому

очевидно, что расшифровать смещению правильно можно только с тем же значением от CPUID с коим оно и было зашифровано!

Далее идет работа по умолчанию: чтение контрольного слова через “обвертку”, затем через виртуальный ESP вытаскивается контрольное слово (BCB2C5D6). Байты B2 и C5, будут одинаковы для всех верхних контрольных слов, которые вытаскивает первый островок. Из этого следует, что по ним можно отождествлять начала всех имеющихся виртуальных стеков. Кроме того, они еще говорят, что второй островок будет одинаков для всех вызывающих VM. Как видим, VM извлекает сначала байт D6, затем прибавляет его к ресурсу (+10), тем самым подготовив следующий ROL-байт для второго островка. Байт C5 будет использован вместе с текущим контрольным байтом (95h), чтобы получить смещение в хранилище №1.5. Над ними первоначально будет произведена операция сложения, затем результат увеличивается в 4 раза, полученное смещение будет сложено с 01FD1000. Прерывает этот процесс прибавление к ресурсу (+4) 4 байт, понятное дело, было использовано только одно контрольное слово, и наш виртуальный указатель сместился вниз (вверх по старшим адресам), тем самым подготовив адрес 1557420h для второго островка. Одним словом, типичный POP EAX.

Возвращаемся к прерванному. VM осталось запросить в хранилище №1.5 зашифрованное смещение второго островка. Для декодирования используется связка CPUID (EAX = 1); MOV CL,AL;ROR EDX, CL. Сырое смещение будет сложено с началом VM, полученный RVA будет извлечен из стека в регистр EAX. Следующий прыжок перенесет нас на второй островок. Существует иерархия в использовании регистров. Так ECX закреплен за текущим ROL-байтом, EDX за контрольным словом, EAX за виртуальным указателем или зашифрованным двойным словом (2-Б), EBX за указателем на главное хранилище, EDI как помощник предыдущего, работает с ресурсами. Этим и объясняется наличие в главном хранилище специальной области, служащей для обмена данными непосредственно между островками.

Настоящая работа VM начинается с островка, который первым занес данные в область хранения главного хранилища. Понятное дело, до этого остальные только “крутили” ROL-байт и “съели” несколько контрольных слов из виртуального стека. Следующий третий (он, четвертый и пятый логически взаимосвязаны) островок будет самым интригующим, все-таки алгоритм расшифровки как-никак. Как было упомянуто выше, первым делом, VM нужны два двойных слова в стеке: первое, это зашифрованный адрес и его нужно декодировать(243BBBD6), второе – уже знакомое нам контрольное слово (4A191A68). В контрольном слове помещается всего 4 байта и все заняты. Поэтому нельзя создать хранилище и как следствие ROL-байт для расшифровываемых адресов, для него попросту нет места. Другое дело – 64 разрядный процессор с его 16 регистрами! Контрольное слово могло бы вместить 8 контрольных байт, существенно расширяя границы для новых хранилищ, а удвоенное количество регистров дало бы простор и новые трюки. Обратная сторона медали – разрастание в размерах островков и как следствие,

куда большие потери в скорости работы защищаемого приложения. Но возвратимся обратно в 32 битный мир: следующим делом рассматриваемый островок заботится о следующем, готовя для него новый ROL-байт. Обыденно, но зато дальше идет то, ради чего они и были созданы: ROR DL, CL; ROR EDX, 8. Представьте револьвер, барабан которого рассчитан на 4 пули, первая ассемблерная команда идет на роль бойка, вторая – поворачивает барабан на следующий патрон: над каждым байтом декодируемого адреса, производится циклический сдвиг вправо на величину контрольного байта! Здесь стоит отметить одну особенность - в зависимости от “дальности” сдвига можно получить умножение и получение “зеркального отражения”.

При CL = FC; DL = 24 => 42 , DL = 3B => B3 При CL = 64; DL = 7D => D7

При CL = D7; DL = 21 => 42

На последнем шаге над результатом будет произведен XOR’инг, причем для каждого островка с расшифровщиком ключ одинаков (43E2AB9D). По существу, алгоритм довольно тривиален, но как уже было неоднократно сказано – при расшифровке все опирается на текущий ROL-байт в младшем регистре CL! Итак, расшифровщик превратил закодированное двойное слово в адрес 015000C4, он нам уже знаком по “стрелке”. Далее, от контрольного слова выдергивается первый байт, умножается на 4, и одним словом, в “главное хранилище” (020A7110) заносится результат вышеупомянутого хоровода. Как видите, тезис о том, что островки не связаны регистрами или стеком верен. Последние две операции, комментировать уже бессмысленно. Разве что наш виртуальный стек съедет вверх на законные два двойных слова. Перепрыгиваем к четвертому. Самый догадливые уже поняли, что это также расшифровщик. Результат его работы - 015110F0 перейдя к оному, несложно догадаться, что это и есть адрес назначения нашего CALL 0040A370! Отлично! Адрес будет сдан на хранение (ну еще бы) в ячейке 020A7290.Пятый островок. У нас есть 015110F0 и 015000C4, по операнду которого работает “стрелка”. Что остается сделать? Ну конечно, перевести! И ведь переводит – в контрольное слово вшиты соответствующие байты - извлекаются адреса 020A7290 и 020A7110. Последнее слово за POP DWORD PTR DS:[EDX], которая и осуществляет магическую подмену из базы запросов на требуемую функцию. Таким образом, следующие обращения к 0040A370(015110F0) будут напрямую. Шестой островок. Кажется, что логика его работы не поддается никакому объяснению. Откуда в 020A701C взялся адрес 0022FF6C? Он и был там до этого (ниже узнаете, кто его оставил). Это сохраненная верхушка стека на момент выполнения следующей асм команды этого островка. Проанализировав последние 4 островка, станет понятным ее назначение. Седьмой островок, являющейся очередным, необычным расшифровщиком, который извлекает число 24h. Необычным, если сопоставить декодируемое двойное слово 02482334h и единственную команду для расшифровки XOR EAX, 02482310h, это не что иное как:

AND EAX,000000FF XOR AL, 10h

То, что нужен только последний байт, догадаться нетрудно. Равно как и в стеке по адресу 0022FF90 (0022FF6C + 24h) лежит знакомый 0040365A. С самого начала адрес возврата не имел никакого смысла, поэтому его замена на адрес требуемой процедуры – событие закономерное, причем как последний этап. Адрес отправляется на хранение в ячейку 020A70A0, и мы уже в восьмом, который декодирует и перезаписывает 015110F0 в ячейке 020A7290 - этакая перестраховка. Итоги работы последних трех островков подводит девятый, который извлекает из последних упомянутых ячеек свежеиспеченные адресаты и перезаписывает адрес возврата. После девятого островка начинается стандартная процедура выхода: сброс spin-блокировки, восстановление регистров и флагов процессора, коррекция стека и долгожданный переход в требуемую функцию по адресу 015110F0.

Что касается способа №1A (рисунок 9), то часть вызовов подчиняется процедуре PreStaticInitDebug(0044F4D1). Она разрезана переходниками в стиле PUSH/RET. Плюс прыжки в середину асм инструкций. Вот неполный список вызываемых WinAPI: SetUnhandledExceptionFilter, GetModuleFileNameA, DeleteFileA (\errors.txt), GlobalFree.

```
0044F669  .  FFD0          CALL EAX                                  SetUnhandleExceptionFilter
0044F66B  .  68 00020000   PUSH 200
0044F670  .  8D85 00FEFFFF LEA EAX,[EBP-200]
0044F676  .  50            PUSH EAX
0044F677  .  56            PUSH ESI
0044F678  .  68 0E284F6F   PUSH 6F4F280E
0044F67D  .  68 E1114000   PUSH 004011E1
0044F682  .  9C            PUSHFD
0044F683  .  817424 08 4E( XOR DWORD PTR SS:[ESP+8],6F9B674E
0044F68B  .  816C24 08 4E( SUB DWORD PTR SS:[ESP+8],6F9B674E
0044F693  .  817424 08 6A: XOR DWORD PTR SS:[ESP+8],553B276A
0044F69B  .  817424 08 6A: XOR DWORD PTR SS:[ESP+8],553B276A
0044F6A3  .  814424 08 4E( ADD DWORD PTR SS:[ESP+8],6F9B674E
0044F6AB  .  9D            POPFD
0044F6AC  .  58            POP EAX
0044F6AD  .  870424        XCHG DWORD PTR SS:[ESP],EAX
0044F6B0  .  FFD0          CALL EAX                                  GetModuleFileNameA
0044F6B2  .  8D85 00FEFFFF LEA EAX,[EBP-200]
0044F6B8  .  6A 5C         PUSH 5C
0044F6BA  .  50            PUSH EAX
0044F6BB  .  68 CFF64400   PUSH 0044F6CF
0044F6C0  .  FF35 6055A20( PUSH DWORD PTR DS:[<&MSVCR80.strrchr>]
0044F6C6  L. C3            RETN                                      RET is used as a jump
```

Рисунок 9 – Способ №1A

За строку “\errors.txt” хотелось бы отметить особо, кроме “стрелок”, для некоторых ASCIIZ строк VM также восстанавливает адреса, а в отдельных случаях – числовые значения(небольшие величины). По аналогии с первым, в виртуальном стеке хранится адрес, который указывает на смещение WinAPI, в таблице импорта игрушки. Из смещения и берется точка входа в процедуру.

Способ №2 (Рисунок 10). Имеет совсем иные задачи в отличие от вышерассмотренных и является частью одного механизма. Вызовы по адресам (0152256Bh и 01520C38) и процедуры, в которых они находятся, включительно

те, которые вызывают. Если смотреть общим планом, то все две работают со вторым аргументом по смещению [EBP+0xC].

Здесь все завязано на Odd-Even Based Cryptography (четная-нечетная базовая криптография).

В первой заложен прямой ход алгоритма, во второй - обратный. Принцип его работы, с помощью логической операции AND которая выполняется над аргументом сначала с маской 5CAC5AC5, затем маска переворачивается (NOT) и логическое И выполняется теперь с маской A353A53A (если выполнить XOR над двумя половинками, то результат – первоначальный вид аргумента). Причем, первая половинка хранится в двух экземплярах. Результат в регистре EAX. К примеру, аргументом будет 0790A442, тогда у нас будет его две половинки 04800040 и 310A402. Насколько я понял, из обоих посредством магических констант(ключей) вытаскиваются два числа 0 и 1, и каждое прибавляется к двух экземплярам первой половинки (04800040 и 04800041). Результирующей операцией XOR получаем 1. Обратный ход по адресу 01520C38 делает все с точностью, наоборот – из 1 и смещений 0 и 1 получает 0790A442. Обращаю твое внимание на закономерность, что есть прямая зависимость от четности/нечетности чисел и чередование значений в 3м столбце.

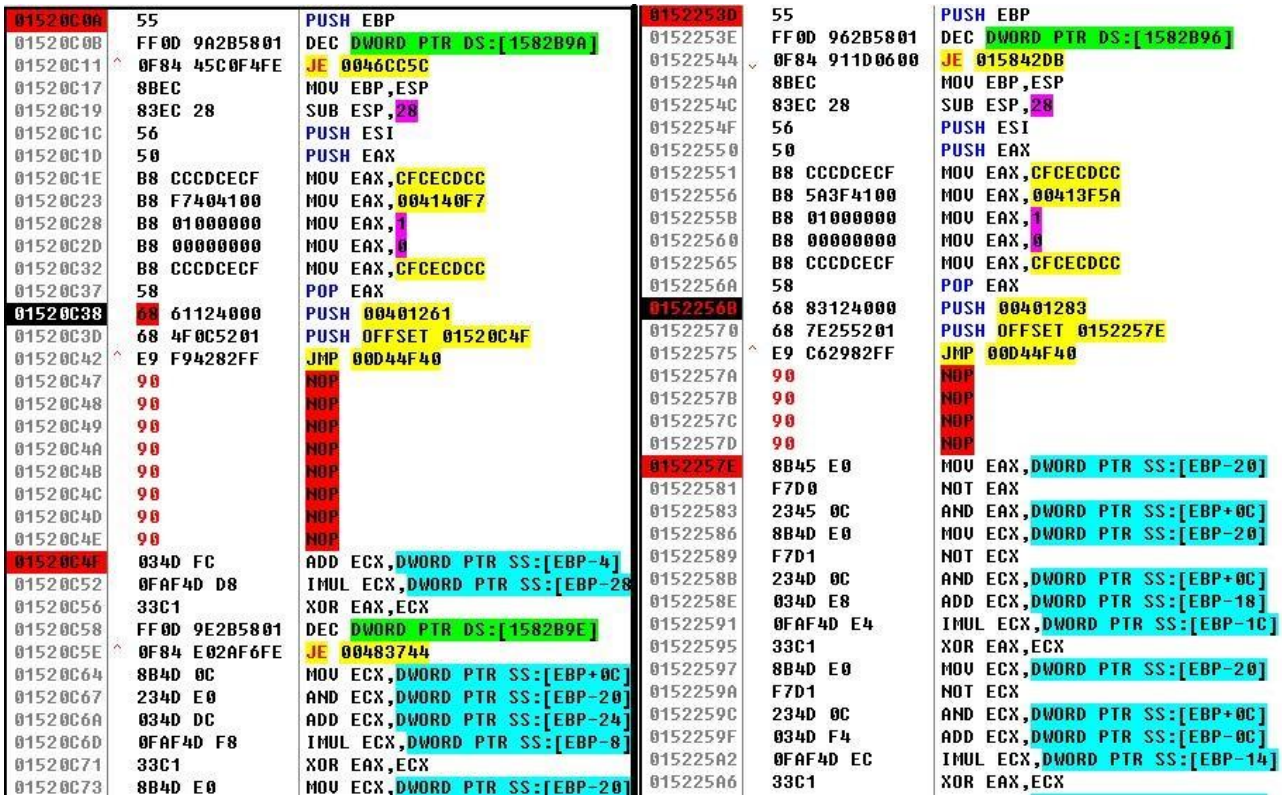

Рисунок 10 – Способ №2

Магические константы лежат по адресу 00B93AFC, оба вызова копируют их в стек, причем первый дополнительно содержит в себе:

MOV ECX, DWORD PTR SS:[ESP+34] //00B93AFC MOV ECX, DWORD PTR DS:[ECX] //5CAC5AC5

MOV EAX, DWORD PTR SS:[ESP+38] //сам аргумент AND EAX, ECX //или AND EAX, 5CAC5AC5

Поэтому сам способ №2 по факту скрывает от глаз несколько асм строк, которые являются частью алгоритма, расположенного ниже (по аргументу адреса возврата).

Кстати, в протоколе трассировки его протяженность составляет чуть более 30 000 строк, это объясняется тем, что островки работают в цикле, копируя из адреса в адрес. У первых двух эта цифра в 10 раз ниже!

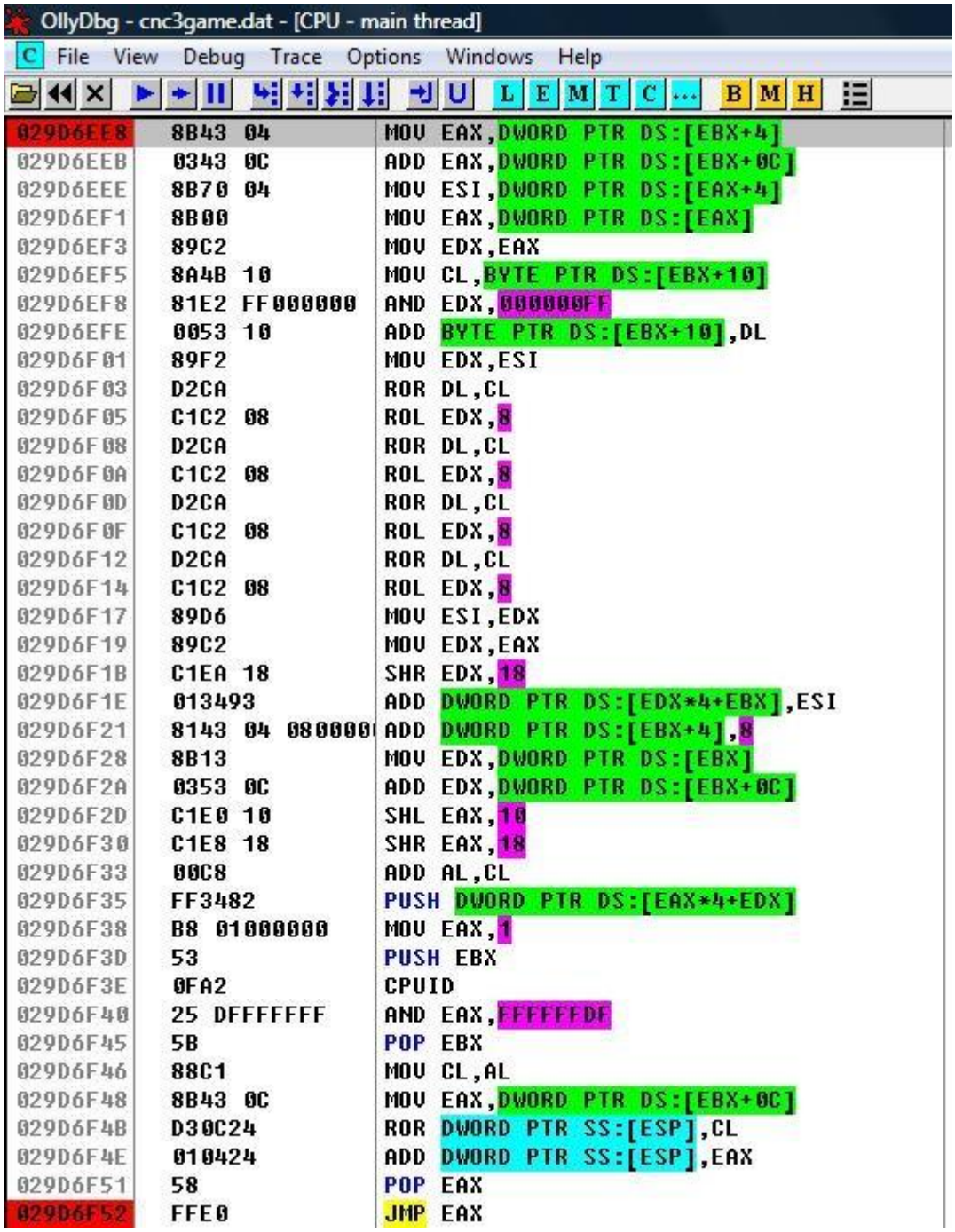

Рисунок 11 – Способ №2

Отдельного разговора заслуживают островки выхода из VM. Способ №1 и №1A используют идентичный вариант выхода:

```
ADD EBX,24
MOV DWORD PTR DS:[EBX],0
POPFD
POPAD
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
PUSH 0
LEA ESP,[ESP+30]
RETN 4
```

В VM существует иерархия в использовании регистров, то EBX = началу главного хранилища. По смещению +24 в главном находится сторожевое число. Отсюда следует, что происходит сброс SPIN блокировки. Логический итог – теперь с VM может работать любой другой поток, что недвусмысленно указывает, что скоро мы покинем виртуальные владения виртуальной машины. Нашу гипотезу подтверждают две последующие инструкции восстановления флагов и регистров процессора. Далее идет типичное заметание следов работы VM, т.е. 12 раз в стек будет положен ноль, затерты 48 байт (30h). В конечном итоге, верхушка съедет обратно на место, где она была до первого PUSH 0 и долгожданный RET 4. Фактически возврат происходит по второму аргументу, который был подменен во всех случаях, кроме вызова от SEH-обработчика (первый – от базы запросов убирается). Как ни странно, но по выходу способ №2 опять выделяется среди первых двух, можно даже сказать – идет впереди, используя прыжки в середину команд:

```
JMP SHORT 2
JMP SHORT 0A
JMP SHORT 1
POP EAX
JMP SHORT 9
JMP SHORT 6
PUSH DWORD PTR DS:[EBX+8]
JMP SHORT 0
```

PUSH 5773
XOR DWORD PTR SS:[ESP],00005757
ADD EBX,DWORD PTR SS:[ESP] ADD ESP,4
JMP SHORT 2
MOV DWORD PTR DS:[EBX],0
JMP SHORT 8
POPFD
JMP SHORT E
POPAD
JMP SHORT 2 RET 4

Т а б л и ц а 3.1 – Смещение по ячейкам

Номер способа	Контрольный байт	Смещение от начала (по ячейкам)
1	B6	2DF
1A	83	20C
2	BF	2FC

Заключительным этапом будет перестроение в нормальный вид секции кода программы. В том числе: замена

PUSH 01234567
RET

Непосредственно на JMP 01234567;
удаление посторонних внедрений типа:

0044F4D2 DEC DWORD PTR DS:[158297A]
0044F4D8 JE NODVD.00482DE5
00482DE5 MOV DWORD PTR DS:[158297A],18D
00482DEF JMP NODVD.0044F4DE

замена вызовов через “стрелку” на непосредственные, перенос процедур и адресов переменных из секции .secuROM в .text и .data. Кстати, используйте олькин Call Tree (Ctrl+K), на последнем этапе вещь незаменимая.

Переход на OEP действительно осуществляется из VM – считая от первого SEH-обработчика на UD2, который естественно передает управление в VM, на 52 вызове, конечно же, по способу №1 управление передается на оригинальную точку входа. В главном хранилище ее адрес(0040A2C7) будет находиться в ячейке по смещению A0. Теперь напоминаю о секрете как попасть ювелирно в OEP. Ошибка умов из SONY DADC очевидна – островок выхода из VM в хранилище №1.5. Тут есть два фактически идентичных варианта для действий:

1) Мы подменяем адрес островка выхода на свой X-код, который ожидает первый адрес возврата в секцию кода. Тут интересно будет упомянуть, что кроме адреса возврата есть и другие приметы, например, ROL–байт равен EAh. Явный референс в сторону Electronic Arts. Вообще скажу по этому поводу только одно: Разработчики SecuROM - явно люди с хорошим чувством юмора и это радует!

Единственный нюанс – правильно посчитать закодированное значение начального адреса нашего перехватчика для записи в таблицу Хранилища №1.5 и перезаписать.

2) Модифицировать непосредственно островок, внедрив туда переход или непосредственно вписав код перехватчика.

Здесь есть один подводный камень для записи кода перехватчика – для островка выхода виртуальной машиной изначально не предусмотрена отдельная выделенная область памяти, поэтому он всегда расположен в толще другого кода или нередко в самом конце выделенной памяти. Т.е. места для всего кода перехватчика может не хватить. Стоит взять на заметку факт, что нередко островки для выхода способов №1 и №1A расположены близко.

Собственной персоной секция .ars – именно код, расположенный в ней, отвечает за создание виртуальной машины SecuROM 7 (WinAPI VirtualAlloc, REPNE MOVSB из секции .securom и XOR BYTE PTR DS:[EAX],CL). Дополнительно стоит отметить еще одну аномальную вещь: по адресу 00C93AE3 вызывается VM.

```
00C93ABA   4A              DEC EDX
00C93ABB   4A              DEC EDX
00C93ABC   75 2A           JNE SHORT 00C93AE8
00C93ABE   C745 C4 985CAE  MOV DWORD PTR SS:[EBP-3C],41AE5C98
00C93AC5   EB 10           JMP SHORT 00C93AD7
00C93AC7   C745 C4 FFA18E  MOV DWORD PTR SS:[EBP-3C],938EA1FF
00C93ACE   EB 07           JMP SHORT 00C93AD7
00C93AD0   C745 C4 39BD1A  MOV DWORD PTR SS:[EBP-3C],221ABD39
00C93AD7   A1 F0292101     MOV EAX,DWORD PTR DS:[12129F0]
00C93ADC   8945 C0         MOV DWORD PTR SS:[EBP-40],EAX
00C93ADF   8D45 C0         LEA EAX,[EBP-40]
00C93AE2   50              PUSH EAX
00C93AE3   E8 58140B00     CALL 00D44F40
00C93AE8   EB 07           JMP SHORT 00C93AF1
00C93AEA   2222            AND AH,BYTE PTR DS:[EDX]
00C93AEC   2200            AND AL,BYTE PTR DS:[EAX]
00C93AEE   06              PUSH ES
00C93AEF   0001            ADD BYTE PTR DS:[ECX],AL
00C93AF1   7E 02           JLE SHORT 00C93AF5
00C93AF3   EB 69           JMP SHORT 00C93B5E
00C93AF5   83EC 14         SUB ESP,14
```

Рисунок 12 – Вызов VM

Однако как-то необычно вызывается (по Способу №1) –управление передается на следующую инструкцию короткого перехода, причем передается как обычно - через адрес возврата положенный вызовом, т.е. в ячейках главного хранилища в конце он не фигурирует! К слову сказать, на этом и подорвался мой X-код, грабящий адреса. Если за’NOP’ипть вызов, то игрушка будет функционировать без ошибок. Предположение о том, что существуют фейковые вызовы VM, подтверждает факт разбора этого вызова: сначала он заносит число 10h в ячейку и занимается его декрементом, причем в ячейке оно

одно, отсюда следует, что из других операций VM, она больше не может ничего с ним сделать! Операция декремента повторяется несколько раз с разными числами до нуля в ячейке. Затем далее идут другие бессмысленные значения и операции, например как чтение ячейки в Хранилище №1.5 со смещением +400 (контрольный байт FFh), в которой естественно ноль! В ячейках главного хранилища фигурирует также адрес процедуры, в которой находится вызов – 00C9A350. К тому весь этот код расположен в секции .ars, а не в .secuROM что не характерно для способа №1. Кстати в .ars расположены SEH для UD2, да и вообще по виду процедура больше походит на проверочный код, что в навесной броне защиты (например, проверка TF). Обобщая только что сказанное, скажу одно – не надо ожидать, что разработчики всегда делают абсолютно все под один шаблон.

Достаточно очевидным является и другое правило, что базы запросов должны всегда совпадать с кодом игрушки, к которому они приставлены! Но ведь и недвусмысленно ясно, что базы запросов в способах №1 и 1-A можно менять местами! И хотя действительно VM выполнит операции руководствуясь подмененной базой (более точнее: виртуальным стеком), в конечном результате запрашиваемая процедура/асм команда будет из подмененной базы, что не сулит ничего хорошего для кода, оперирующего запрашиваемой процедурой. Ну а если опять же вспомнить, что в некоторых базах восстанавливаются адреса и начальные переменные, тут уж тем более жди вагон необрабатываемых исключений.

Было представлено серьезное кумулятивное оружие для уничтожения основной навесной брони SecuROM v7, которое основывается на просчетах разработчиков защитного комплекса. Конечно это не ноу-хау в полном смысле, а скорее всего попытка привести в нормальный вид имеющуюся теорию и показать на реальном примере работу нашего кумулятивного снаряда. Каким бы страшным изначально не казался SecuROM v7 с его VM, в конце концов, останется один факт – все действительно просто! Хорошая новость для тех, кто хочет дисассемблировать протектор в OllyDbg v2.0, забыв про все фантомы и другие подобные стелс-плагины. Невероятно, но до 7.40 этот нехитрый и вполне очевидный трюк работает (вспомни, что только при условии: вместо ollydbg.exe будет elf.exe… короче все что угодно, но только не оригинальное имя исполняемого файла отладчика):

```
MOV ESI,DWORD PTR FS:[18]      ; < НОВАЯ ТОЧКА ВХОДА
MOV ESI,DWORD PTR DS:[ESI+30]
MOV DWORD PTR DS:[ESI],0       ; АКТИВИРУЕМ КОМПЛЕКС "НАКИДКА"
PUSH 011DBA77 ; ASCII "NTDLL"
MOV EAX,DWORD PTR DS:[<&KERNEL32.GetModuleHandleA>]
CALL EAX
PUSH 011DBA7E; ASCII "NtQueryInformationProcess"
PUSH EAX
MOV EAX,DWORD PTR DS:[<&KERNEL32.GetProcAdress>]
```

```
CALL EAX
MOV ESI,DWORD PTR DS:[<&KERNEL32.VirtualProtect>]
PUSH 011DBA9A
PUSH 40
MOV EDI,EAX
PUSH 100
PUSH EAX
CALL ESI
ADD EDI,6
MOV EAX,011DB8B7
XCHG DWORD PTR DS:[EDI],EAX ;подмена вызова KiFastSystemCall
SUB EDI,6
MOV DWORD PTR DS:[11DB8BB],EAX
MOV AX,40
MOV WORD PTR DS:[11DBA9A],AX
PUSH 011DBA9A
PUSH 20
PUSH 100
PUSH EDI
CALL ESI
XOR EDI,EDI
MOV ESI,EDI
JMP 011D7B30 ; ВОВРАЩАЕМСЯ НА ПРЕДЫДУЩУЮ ТОЧКУ ВХОДА
011DB8B7  C0B8 1D010003F
011DB8BE  7F 00
MOV ECX,DWORD PTR SS:[ESP+0C] ;КОД НАШЕГО ПЕРЕХВАТЧИКА MOV CH,7 ; if(ProcessInfoClass == ProcessDebugPort)…
SUB CH,CL
JNE SHORT 011DB8D8
XOR EAX,EAX
MOV ECX,DWORD PTR SS:[ESP+10]
MOV DWORD PTR DS:[ECX],EAX ADD ESP,4
RETN 14
```

Но прежде всего, хочу обратить внимание на WinAPI VirtualProtect. Внедрение X-кода строилось на необходимости получить доступ на запись в секцию кода. Использование VirtualProtect – самый правильный способ изменить атрибуты страницы, в данном случае атрибут защиты меняется с PAGE_EXECUTE_READ(20h) на PAGE_EXECUTE_READWRITE(40h) и обратно.

4 Технико-экономическое обоснование

Технико-экономическое обоснование научно-исследовательских работ проводится с целью определения и анализа трудовых и денежных затрат, направленных на их реализацию, а также уровня их научно-технической результативности.

Целью дипломной работы является изучение принципа работы популярных технических средств защиты авторских прав (DRM-защиты), методы их взлома, сильные и слабые стороны, а также их негативное влияние на систему и пользователей. Исследование также затронет целесообразность внедрения такой защиты на предметы авторского права.

4.1 Определение трудоемкости выполнения НИР

Для определения трудоемкости выполнения НИР прежде всего был составлен перечень всех основных этапов и видов работ, которые должны быть выполнены. При этом особое внимание было уделено логическому упорядочению последовательности отдельных видов работ и выявлению возможностей их параллельного выполнения, что позволило существенно сократить общую длительность проведения НИР.

Форма разделения работ по этапам с указанием трудоемкости их выполнения приведена в таблице 4.1.

Таблица 4.1 – Распределение работ по этапам и видам и оценка их трудоемкости

Этап проведения НИР	Вид работы на данном этапе	Трудоемкость выполнения НИР, чел.× ч.
1	Постановка задачи	22
2	Разработка и утверждение технического задания (ТЗ)	30
3	Подбор и изучение материалов по тематике	350
4	Анализ актуальных DRM-защит	40
5	Пример взлома	70
6	Оформление работы	60
7	Подведение итогов	50
ИТОГО трудоемкость выполнения дипломной работы		622

4.2 Расчет затрат на выполнение НИР

Определение затрат на выполнение НИР производится путем составления соответствующей сметы, которая включает следующие статьи:

1) Материальные затраты.

2) Затраты на оплату труда.

3) Отчисления на социальные нужды.
4) Амортизация основных фондов.
5) Прочие затраты.

4.2.1 Расчет материальных затрат

В статью «Материальные затраты» включаются затраты на основные и вспомогательные материалы, энергию, необходимые для выполнения НИР.

Расчет затрат на материальные ресурсы производится по форме, приведенной в таблице 4.2.

Т а б л и ц а 4.2 – Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Листы А4	упаковка	4	950	3 800
Листы А3	упаковка	1	2 300	2 300
USB накопитель	шт	3	2 000	6 000
DVD-диск	шт	5	200	1 000
Картридж	шт	2	2 000	4 000
Ручки	упаковка	5	100	500
ИТОГО затраты на материальные ресурсы				17 600

Расчет затрат на оборудование и программное обеспечение производится по форме, приведенной в таблице 4.3.

Т а б л и ц а 4.3 – Затраты на оборудование и программное обеспечение

Наименование	Единица измерения	Количество	Цена за единицу, тг	Сумма, тг
Игровая приставка XBOX360	шт	1	90 000	90 000
Игровая приставка PlayStation3	шт	1	120 000	120 000
Ноутбук HP Pavilion g6	шт	1	99 000	99 000
Microsoft Windows 10	лиц. копия	1	24 000	24 000
Microsoft Office 2016	лиц. копия	1	26 000	26 000
Видеоигры	лиц. копия	2	2 000	4 000

ИТОГО затраты на материальные ресурсы	363 000

Общая сумма затрат на материальные ресурсы ($З_М$) определяется по формуле:

$$З_м = \sum_{i=1}^{n} Р_i \cdot Ц_i ,$$

(4.1)

где $Р_i$ – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;

n – количество видов материальных ресурсов.

Итого общая сумма затрат на материальные ресурсы, оборудование и программное обеспечение составляет 380 600 тг.

4.2.2 Расчет затрат на электроэнергию

Так как при выполнении НИР использовалось электрооборудование, то необходимо было рассчитать затраты на электроэнергию

Расчет затрат на электроэнергию производится по форме, приведенной в таблице 4.2.

Т а б л и ц а 4.2 – Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для выполнения НИР, ч	Цена электроэнергии, $\frac{тенге}{кВт*ч}$	Сумма, тг
Ноутбук Lenovo Ideapad Y570	0,062	0,9	240	27	362
Ноутбук HP Pavilion g6	0,047	0,9	240	27	274
МФУ HP LaserJet M125a	0,465	0,9	80	27	904
Монитор ASUS	0,046	0,9	240	27	268
Игровая приставка XBOX 360	0,15	0,9	10	27	36
Игровая приставка Playstation 3	0,2	0,9	14	27	68
ИТОГО затраты на электроэнергию					1 912

Общая сумма затрат на электроэнергию ($З_Э$)рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} М_i \cdot К_i \cdot Т_i \cdot Ц\,, \qquad (4.2)$$

где $М_i$ – паспортная мощность i-го электрооборудования, кВт;

$К_i$ – коэффициент использования мощности i-го электрооборудования (принимается $К_i$=0.7; 0.9);

$Т_i$ – время работы i-го оборудования за весь период выполнения НИР, ч;

Ц – цена электроэнергии, тг/кВт×ч.;

i – вид электрооборудования;

n – количество электрооборудования.

4.2.3 Расчет затрат на оплату труда

В статью «Затраты на оплату труда» включаются расходы по оплате труда всех работников, занятых выполнением НИР (дипломника, руководителей и консультантов дипломной работы, привлеченных лиц).

Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.4.

Т а б л и ц а 4.4 – Затраты на оплату труда

Категория работника	Месячная заработная плата, тг	Часовая ставка, тг/ч	Трудоемкость выполнения НИР, чел.×ч	Сумма, тг
Инженер	50 000	306	472	144 432
Научный руководитель	80 000	490	150	73 500
ИТОГО затраты на оплату труда				217 932

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \cdot Т_i\,, \qquad (4.3)$$

где $ЧС_i$ – часовая ставка i-го работника, тг;

$Т_i$ – трудоемкость выполнения НИР, чел.×ч;

i – категория работника;

n – количество работников, занятых выполнением НИР.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \qquad (4.4)$$

где $ЗП_i$ – месячная заработная плата i-го работника, тг

$ФРВ_i$ – месячный фонд рабочего времени i-го работника, $ФРВ_i$ = 163,36 ч

4.2.4 Расчет отчислений на социальные нужды

Затраты по этой статье составляют отчисления по единому социальному налогу (ЕСН).

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 11 % от ФОТ (фонда оплаты труда). Следует отметить, что пенсионные отчисления не облагаются социальным налогом. Отчисления по заработной плате определяются по следующей формуле:

$$Ос = (ФОТ - ПО) \cdot 0{,}11, \quad (4.5)$$

где ПО - отчисления в пенсионный фонд, что составляет 10% от ФОТ,

$$ПО = 217\,932 \cdot 0{,}1 = 21\,793 \text{ тг.}$$

Итак, отчисления из заработной платы составили:

$$Ос = (217\,932 - 21\,793) \cdot 0{,}11 = 21\,575 \text{ тг.}$$

4.2.5 Расчёт амортизационных отчислений

В статью «Амортизация основных фондов» включается сумма амортизационных отчислений от стоимости оборудования и приборов, используемых при выполнении НИР. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 4.5.

Общая сумма амортизационных отчислений определяется по формуле:

$$З_м = \sum_{i=1}^{n} \frac{Ф_i \cdot Н_{Ai} \cdot Т_{НИРi}}{100 \cdot Т_{Эфi}}, \quad (4.6)$$

где $Ф_i$ – стоимость i-го оборудования, тг;

$Н_{Ai}$ – годовая норма амортизации i-го оборудования, %;

$Т_{НИРi}$ – время работы i-го оборудования за весь период выполнения НИР, ч;

$Т_{Эфi}$ – эффективный фонд времени работы i-го оборудования за год, ч/год;

i – вид оборудования;

n – количество оборудования.

Приведем расчеты для каждого оборудования:

1) Ноутбук Lenovo Ideapad Y570

$$З_м = \frac{165\,000 \cdot 20 \cdot 240}{100 \cdot 2416} = 3\,278 \text{ тг.}$$

2) Ноутбук HP Pavilion g6

$$З_м = \frac{99\,000 \cdot 20 \cdot 240}{100 \cdot 2\,416} = 1\,967 \text{ тг.}$$

3) МФУ HP LaserJet M125a

$$З_м = \frac{43\,000 \cdot 20 \cdot 80}{100 \cdot 759} = 906 \text{ тг.}$$

4) Монитор ASUS

$$З_м = \frac{100 \cdot 20 \cdot 240}{100 \cdot 1\,518} = 3\,162 \text{ тг.}$$

5) Игровая приставка XBOX360

$$З_м = \frac{90\,000 \cdot 20 \cdot 10}{100 \cdot 1\,518} = 119 \text{ тг.}$$

6) Игровая приставка PlayStation3

$$З_м = \frac{120\,000 \cdot 20 \cdot 14}{100 \cdot 1\,518} = 221 \text{ тг.}$$

Итого общая сумма амортизационных отчислений составит 9653 тг.

Т а б л и ц а 4.5 – Амортизация основных фондов

Наименование оборудования	Стоимость оборудования, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования, ч/год	Время работы оборудования для выполнения НИР, ч	Сумма, тг
Ноутбук Lenovo Ideapad Y570	165 000	20	2 416	240	3 278
Ноутбук HP Pavilion g6	99 000	20	2 416	240	1 967
МФУ HP LaserJet M125a	43 000	20	759	80	906
Монитор ASUS	100 000	20	1 518	240	3 162
Игровая приставка XBOX 360	90 000	20	1 518	10	119
Игровая приставка Playstation 3	120 000	20	1 518	14	221
ИТОГО амортизация основных фондов					9 653

Годовые нормы амортизации оборудования принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования оборудования:

$$Н_{Ai} = \frac{100}{Т_{Ni}}, \quad (4.7)$$

где $Т_{Ni}$ – возможный срок использования i-го оборудования, год.

Возможный срок полезного использования для всего оборудования составляет 5 лет.

4.2.6 Расчет прочих затрат

В статью «Прочие затраты» включаются расходы на арендную плату, включая коммунальные платежи, расходы на рекламу, канцелярские и прочие хозяйственные расходы.

Затраты на арендную плату определяются в зависимости от стоимости аренды 1 кв. м занимаемой площади.

Т а б л и ц а 4.6 – Затраты на арендую плату

Площадь, кв.м.	Цена за кв.м., тг	Цена за месяц, тг	Срок, месяц	Сумма, тг
19,5	3 000	58 500	3	175 500

Затраты на оплату интернета берутся на основании стоимости услуг, предоставляемых оператором, тарифов и срока использования данного пакета услуг. При выполнении работы был использован тариф ID Net Turbo, его стоимость и расчёт общей суммы указан в таблице 4.7.

Т а б л и ц а 4.7 – Затраты на пользование интернета

Цена за месяц, тг	Срок, месяц	Сумма, тг
4 600	3	13 800

Итого прочие затраты составляют 189 300 тг

4.2.7 Составление сметы

На основании полученных данных по отдельным статьям была составлена смета затрат за выполнение НИР по форме, приведенной в таблице 4.8.

Т а б л и ц а 4.8 - Смета затрат на выполнение НИР

Статьи затрат	Сумма, тг
1. Материальные затраты, в том числе:	
- материалы	380 600
- электроэнергия	1 912
2. Затраты на оплату труда.	217 932
3. Отчисления на социальные нужды.	21 575
4. Амортизация основных фондов.	9 653
5. Прочие затраты.	189 300
ИТОГО по смете	820 972

4.3 Определение возможной (договорной) цены НИР

Величина возможной (договорной) цены НИР должна устанавливаться с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена ($Ц_Д$) для прикладных НИР рассчитывается по формуле:

$$Ц_Д = З_{НИР} \cdot \left(1 + \frac{Р}{100}\right), \quad (4.8)$$

где $З_{НИР}$ – затраты на выполнение НИР (из таблицы 4.8), тг;

Р – средний уровень рентабельности НИР, % (принимается в размере 20-30% по согласованию с консультантом по экономической части).

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2016 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Цр = Цд + Цд \cdot НДС, \quad (4.9)$$

Приняв во внимание все предыдущие расчеты определим возможную (договорную) цену НИР

$$Ц_Д = 820\,972 \cdot \left(1 + \frac{20}{100}\right) = 985\,166 \text{ тг.}$$

Цена с учетом НДС

Цр = 985 166 + 985 166 * 0,12 = 1 103 385 тг.

4.4 Оценка научно-технической результативности и социальной эффективности НИР

В настоящее время данная тема является одной из актуальных, резко критикуемых тем связанных с распространением контента в интернете. С «незаконным» распространение контента в сети борются и на государственном уровне, ограничивая доступ к варез-сайтам и торрент-трекерам. Но пока от этого никто не выигрывает. Данное исследование даст следующие возможности:

- Выявление сильных и слабых сторон даст разработчикам сконцентрироваться на сильных сторонах защиты и избавиться от недостатков. Это в свою очередь повысит защиту контента с авторским правом и минимизирует причиняемый вред системе и пользователям.

- Владельцы авторского права на основе исследования будут грамотно внедрять DRM в свои продукты, или не внедрять их вовсе, а устанавливать доступные цены на свои продукты для регионов с высоким уровнем пиратства. Это позволит получать прибыль, при этом не тратясь на дорогостоящие системы защиты.

- Исследование поможет государству бороться с пиратским контентом не ограничивая свободу пользователей в интернете.

- Обычные люди, то есть пользователи контента не будут ограничены использованием интернета, получат возможность приобретать продукты

авторского права по доступной цене, а DRM-зашита больше не будет вызывать раздражение

5. Безопасность жизнедеятельности

Объектом дипломной работы являются работы, направленные на исследование технических средств защиты авторских прав, а именно:

– анализ и актуализация существующих средств;

– взлом DRM защиты методом реверсинга.

Описанные выше работы проводятся в помещении расположенные в корпусе “Д” Алматинского Университета Энергетики и Связи на кафедре «Компьютерных технологий» в аудитории 306. Аудитория оснащена компьютерной техникой:

– персональные компьютеры (ПК) – под ПК будем понимать совокупность из монитора, системного блока, клавиатуры, мыши и проводов для подключения описанных выше устройств;

– проекторы;

– столы и стулья;

Данное помещение относиться к классу помещению без повышенной опасности, так как отсутствуют условия, создающие повышенную или особо повешенную опасность.

Также данное помещение оснащено оборудованием противопожарной сигнализацией и датчиками дыма.

Согласно требованиям СанПиН 2.2.2/2.4.1340–03, расстояние между рабочими столами с видеомониторами, равно 2 м, а расстояние между боковыми поверхностями видеомониторов примерно 1,2 м. Площадь на одно рабочее место пользователей ПК с монитором на базе плоских дискретных экранов (жидкокристаллические, плазменные) - 4,5 $м^2$ [24].

5.1 Анализ потенциально опасных и вредных факторов

5.1.1 Анализ выявленных вредных факторов проектируемой производственной среды

При выполнении работ на персональном компьютере (ПК) согласно ГОСТу 12.0.003-74 “ССБТ. Опасные и вредные производственные факторы. Классификация” могут иметь следующие факторы[16]:

– повышенная температура поверхностей ПК;

– пониженная или повышенная температура воздуха на рабочем месте;

– пониженная влажность воздуха;

– действие статического электричества;

– повышенный уровень электромагнитных излучений;

– отсутствие или недостаток естественного света;

– недостаточная искусственная освещённость рабочей зоны;

– монотонность трудового процесса;

– зрительное напряжение.

Все выше перечисленные факторы негативно влияют на организм человека и на его здоровье.

Теперь рассмотрим все факторы подробнее.

Повышенная температура поверхностей ПК, пониженная влажность воздуха, пониженная или повышенная температура воздуха на рабочем месте пагубно воздействует организм человека. А также, высокая температура благотворно влияет на рост бактерий, что влечет риск возникновения разного рода заболеваний.

Пониженная влажность весьма плохо сказывается на здоровье. Пребывание в помещении нарушает естественный баланс влажности. И если в летнее время это может быть незаметным, то зимой разница показателей относительной влажности на улице и в помещении становится более заметной. Это объясняется тем, что уровень относительной влажности уличного воздуха понижается при его нагреве системой отопления. Колебания влажности воздуха, как в сторону уменьшения, так и в сторону увеличения, негативно влияют на самочувствие и здоровье. Это может провоцировать различные недомогания, головные и физические боли, снижение иммунитета, может появиться чувство усталости, неуюта, упадок сил, нежелание работать. Организм человека незамедлительно реагирует на снижение влажности воздуха - из тела с повышенной скоростью начинает испаряться влага. Взаимодействие с сухим воздухом, в первую очередь, проявляется в ощущении сухости слизистой оболочки носа и дыхательных путей, сухости кожи (рук и лица), пересыхания губ. Избыточная сухость воздуха провоцирует раздражение носа, ангину, обветривание кожи и губ, может привести к проблемам с дыханием. Избыточная влажность воздуха также неблагоприятна: может вызвать аллергические реакции, астму, ринит.

Согласно требованию СанПиН 2.2.2/2.4.1340–03, в офисе поддерживается температура равная 19 – 20 С°, при относительной влажности в 55 – 58 %. Оптимальные параметры микроклимата в офисах приведены в таблице 5.1, согласно СанПиН 2.2.2/2.4.1340–03 [15].

Таблица 5.1 – Оптимальные параметры микроклимата помещений с использованием ПК

Температура, С°	Относительная влажность, %	Абсолютная влажность, г/м^3	Скорость движения воздуха, м/с
19	62	10	<0,1
20	58	10	<0,1
21	55	10	<0,1

Статическому электричеству присуще свойство накапливаться в человеке, что ведет к таким проблемам как раздражительность, плохой сон, психологическим заболеваниям, склонность к артериальной гипертензии.

Электромагнитное излучение приводит к биохимическим изменениям, происходящих в клетках и тканях человека. Особое воздействие оказывается на нервную и сердечнососудистую систему человека. Так же возможны отклонения со стороны эндокринной системы человека. Это влияет на общее состояние человека, повышается возбудимость нервной системы, проявляется эмоциональная неустойчивость.

Оценка величины уровней электромагнитного поля (ЭМП), проведенная по паспортным данным компьютера и монитора, показала их соответствие нормам ТСО–03 и СанПиН 2.2.2/2.4.1340–03 “Гигиенические требования к персональным электронно-вычислительным машинам и организации работы” [15]. В таблице 5.2 приведены нормы уровня ЭМП, которым соответствует техника в офисе.

Недостаток освещенности рабочего места утомляет не только зрение, но и вызывает утомление организма в целом. Неправильное освещение может быть причиной травматизма: плохо освещенные опасные зоны, слепящие лампы, резкие тени ухудшают или вызывают полную потерю зрения, ориентации.

Т а б л и ц а 5.2 – Допустимые уровни ЭМП, создаваемых ПК

Наименование параметров		ВДУ ЭМП
Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Электростатический потенциал экрана видеомонитора		500 В

Освещенность на рабочем месте должна соответствовать гигиеническим нормам. Увеличение освещенности рабочей поверхности до определенного предела улучшает видимость объекта, увеличивает скорость различения предметов и повышает производительность труда. Освещенность на поверхности стола в зоне размещения рабочего документа равна 300 - 500 лк. Освещение не создает бликов на поверхности экрана. Освещенность поверхности экрана равна примерно 300 лк. Яркость на рабочей поверхности и в пределах окружающего пространства должна распределяться по возможности равномерно, так как при переводе взгляда с ярко освещенной на слабо освещенную поверхность и наоборот глаз должен адаптироваться, что вызывает его утомление. Равномерному распределению яркости способствует светлая окраска потолка, стен, оборудования. Яркость светильников общего освещения в зоне углов излучения от 50 до 90 градусов с вертикалью в продольной и поперечной плоскостях составляет не более 200 кд/м^2. Все описанные показателя соблюдают нормы СанПиН 2.2.2/2.4.1340–03 [15].

Теперь рассмотрим основные аспекты длительной работы за ПК.

Работающий за компьютером человек длительное время должен сохранять относительно неподвижное положение, что негативно сказывается на позвоночнике и циркуляции крови во всем организме (застой крови).

Чтение информации с монитора вызывает перенапряжение глаз. Возникает это главным образом потому, что во время чтения с монитора расстояние от текста до глаз постоянно остается одним и тем же, из-за этого мышцы глаз, регулирующие аккомодацию, находятся в постоянном напряжении. Со временем это может привести к нарушению аккомодативной способности глаз и, следовательно, к нарушениям зрения.

Человек, работающий за компьютером, вынужден все время принимать решения, от которых зависит эффективность его работы. Порой бывает довольно сложно предположить последствия того или иного шага (особенно на фоне хронической усталости). Поэтому, длительная работа за компьютером, часто является причиной хронического стресса.

Мониторы, снабженные электронной пушкой, являются сильным источником электромагнитных полей. Постоянная «бомбардировка» организма человека ускоренными электронами приводит к различным расстройствам нервной системы и глаз.

Работа за компьютером предполагает переработку большого массива информации и постоянную концентрацию внимания, поэтому при длительной работе за компьютером нередко нарушается внимание. Вредные факторы, описанные и охарактеризованные выше, в рабочих помещениях контролируются различными нормами, которые накладывают количественные изменения.

При работе за компьютером нужно соблюдать следующие нормы:

– пространство для ног (ширина не менее 500 мм);

– высота рабочей поверхности, при организации рабочего места 680 мм;

– высота сиденья 420 мм;

– очень часто используемые средства отображения информации, требующие точного и быстрого считывания показаний, следует располагать в вертикальной плоскости под углом +- 15° от нормальной линии взгляда и в горизонтальной плоскости под углом +- 15° от сагиттальной плоскости [15].

Более подробно сведения о размерности стульев и столов описаны в СанПиН 2.2.2/2.4.1340–03 “Гигиенические требования к персональным электронно-вычислительным машинам и организации работы” [15].

В соответствии с допустимыми нормами, которые описаны выше, можно выработать средства коллективной защиты:

– установка ионизаторов воздуха с функциями анализа окружающей среды в офисе;

– в дневное время суток раскрывать окна для достаточного поступления света в помещение, если же в офисе отсутствует достаточное количество оконных проемов, то желательно иметь светло-теплую цветовую гамму в офисе;

– периодически проводить проветривание помещения, производить влажную уборку, а также при необходимости устанавливать увлажнители воздуха в помещении;

– для уменьшения воздействия электростатического поля на организм человека следует устанавливать антистатические поверхности на полу, закупка офисной мебели, которая не провоцирует статическое электричество;

– для контроля над температурой окружающей среды следует устанавливать термостаты, и регулировать температуру в помещении согласно описанным выше нормам.

К индивидуальным средствам защиты можно отнести следующее:

– периодических отдых, гимнастика для глаз и рук;

– периодические прогулки на свежем воздухе.

5.1.2 Анализ выявленных опасных факторов проектируемой производственной среды

Данный раздел описывает различного рода опасные факторы, к которым можно отнести следующее:

– механические опасности;

– термические опасности;

– электробезопасность;

– пожаровзрывобезопасность.

К механическим опасностям, возникающим при работе с ПК в аудитории, можно отнести такие как:

– столы;

– стулья;

– мониторы;

– системные блоки и другое мультимедийное оборудование больших габаритов;

– кондиционеры;

– радиаторы отопления.

При работе с описанными выше предметами, чтобы не было механических травм, необходимо соблюдать простые инструкции:

– не переставлять и не поднимать мебель и оборудование без согласования с руководством;

– отключать оборудование перед перемещением;

– не пытаться починить оборудование самостоятельно.

Термические опасности в аудитории могут быть связаны со следующими причинами:

– источники бесперебойного питания;

– ПК;

– радиаторы отопления.

Во избежание термических опасностей нужно соблюдать следующие правила:

– не сливать воду с радиаторов отопления;

– не разбирать любую технику во включенном состоянии.

Электробезопасность является опасным фактором, так как в помещениях имеются различные электрические приборы и установки, которые могут причинить опасность поражения электрическим током. Обычно они связаны со следующими источниками:

– поражение электрическим током;

– статическое электричество;

– молниезащита.

В целях безопасности при работе в помещении запрещается:

– использовать для подключения нестандартные разъемы;

– проводить ремонт оборудование при включенном состоянии;

– самостоятельно открывать крышку мониторов и системных блоков компьютеров;

– для безопасности во время гроз необходимо удостовериться о наличие молниеотвода, и того факта что все розетки в офисном помещении заземлены.

Пожаровзрывобезопасность характеризуется следующими причинами:

– возгорание на рабочем месте в связи с коротким замыканием;

– возгорание на рабочем месте в связи с неправильным обращением с огнем.

Помещение оснащено пожарной сигнализацией и датчиками дыма.

При невозможности самостоятельно потушить пожар необходимо вызвать пожарную команду, после чего поставить в известность о случившемся инженера по техники безопасности.

5.2 Расчетная часть

5.2.1 Расчет освещенности

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и размещения. Исходя из этого, рассчитаем параметры искусственного освещения.

Обычно искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют ряд существенных преимуществ [18]:

- по спектральному составу света они близки к дневному, естественному свету;

- обладают более высоким КПД (в 1,5-2 раза выше, чем КПД ламп накаливания);

- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);

- более длительный срок службы.

Расчет освещения производится для комнаты площадью 15 м^2, ширина которой 5 м, высота - 3 м. Воспользуемся методом светового потока [18].

Для определения количества светильников определим световой поток, падающий на поверхность по формуле:

$$F = \frac{E \cdot K \cdot S \cdot Z}{n}, \quad (5.1)$$

где F – рассчитываемый световой поток, Лм;

E – нормированная минимальная освещенность, Лк (определяется по таблице). Работу программиста, в соответствии с этой таблицей, можно отнести к разряду точных работ, следовательно, минимальная освещенность будет E = 300 Лк;

S – площадь освещаемого помещения (в нашем случае S = 15 м^2);

Z – отношение средней освещенности к минимальной (обычно принимается равным 1,1…1,2 , пусть Z = 1,1);

K – коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (его значение зависит от типа помещения и характера проводимых в нем работ и в нашем случае K = 1,5);

n – коэффициент использования, (выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп и исчисляется в долях единицы; зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемых коэффициентами отражения от стен ($P_С$) и потолка ($P_П$)), значение коэффициентов $P_С$ и $P_П$ были указаны выше: $P_С$ = 40%, $P_П$ = 60%. Значение n определим по таблице коэффициентов использования различных светильников. Для этого вычислим индекс помещения по формуле:

$$I = \frac{S}{h \cdot (A + B)}, \quad (5.2)$$

где S - площадь помещения, S = 15 м^2;

h - расчетная высота подвеса, h = 2.92 м;

A - ширина помещения, A = 3 м;

B - длина помещения, B = 5 м.

Подставив значения получим:

$$I = \frac{15}{2,92 \cdot (3+5)} = 0,64$$

Зная индекс помещения I, по таблице 7 [23] находим n = 0,22

Подставим все значения в формулу для определения светового потока F:

$$F = \frac{300 \cdot 1,5 \cdot 15 \cdot 1,1}{0,22} = 33750 \; Лм$$

Для освещения выбираем люминесцентные лампы типа ЛБ40-1, световой поток которых F = 4320 Лк.

Рассчитаем необходимое количество ламп по формуле:

$$N = \frac{F}{F_л}, \tag{5.3}$$

где N - определяемое число ламп;

F - световой поток, F = 33750 Лм;

$F_л$- световой поток лампы, $F_л$ = 4320 Лм.

$$N = \frac{33750}{4320} = 8 \; шт.$$

При выборе осветительных приборов используем светильники типа ОД. Каждый светильник комплектуется двумя лампами.

5.2.2 Расчет уровня шума

Одним из неблагоприятных факторов производственной среды в ИВЦ является высокий уровень шума, создаваемый печатными устройствами, оборудованием для кондиционирования воздуха, вентиляторами систем охлаждения в самих ЭВМ.

Для решения вопросов о необходимости и целесообразности снижения шума необходимо знать уровни шума на рабочем месте оператора.

Уровень шума, возникающий от нескольких некогерентных источников, работающих одновременно, подсчитывается на основании принципа энергетического суммирования излучений отдельных источников [19]:

$$L_\Sigma = 10 \lg \sum_{i=1}^{i=n} 10^{0,1 L_i}, \tag{5.4}$$

где L_i – уровень звукового давления i-го источника шума;

n – количество источников шума.

Полученные результаты расчета сравнивается с допустимым значением уровня шума для данного рабочего места. Если результаты расчета выше допустимого значения уровня шума, то необходимы специальные меры по снижению шума. К ним относятся: облицовка стен и потолка зала звукопоглощающими материалами, снижение шума в источнике, правильная планировка оборудования и рациональная организация рабочего места оператора.

Уровни звукового давления источников шума, действующих на оператора на его рабочем месте представлены в табл. 5.3.

Т а б л и ц а 5.3 – Уровни звукового давления различных источников.

Источник шума	Уровень шума, дБ
Жесткий диск	40
Вентилятор	45
Монитор	17
Клавиатура	10
Принтер	45
Сканер	42

Обычно рабочее место оператора оснащено следующим оборудованием: винчестер в системном блоке, вентилятор(ы) систем охлаждения ПК, монитор, клавиатура, принтер и сканер.

Подставив значения уровня звукового давления для каждого вида оборудования в формулу , получим:

$$L_{\Sigma} = 10 \cdot \lg (10^{4} + 10^{4,5} + 10^{1,7} + 10^{1} + 10^{4,5} + 10^{4,2}) = 49,5 \text{ дБ}$$

Полученное значение не превышает допустимый уровень шума для рабочего места оператора, равный 65 дБ (ГОСТ 12.1.003-83). И если учесть, что вряд ли такие периферийные устройства как сканер и принтер будут использоваться одновременно, то эта цифра будет еще ниже. Кроме того, при работе принтера непосредственное присутствие оператора необязательно, т.к. принтер снабжен механизмом автоподачи листов.

Заключение

Защита авторских прав – это достаточно сложное правовое явление, что порождает неоднозначные подходы к его изучению.

В результате проведенного исследования, подвергнув анализу современные технические средства защиты авторских прав можно сделать следующие выводы:

DRM защита является полностью неэффективной. Как показала практика взломать такую защиту достаточно просто, а время взлома может колебаться в пределах нескольких дней и до нескольких месяцев

Внедрение DRM приводит к дополнительным затратам, что в свою очередь и так удорожает стоимость конечного продукта

DRM нагружает операционную систему, из-за чего скорость выполнения защищенного продукта снижается.

DRM создает дыры в защите системы. Так как DRM легко подается взлому, злоумышленникам не составляет труда внести в него вредоносное ПО для получения доступа к системе.

DRM защита выводит из строя аппаратное обеспечение устройств. Работа DRM связана с неоднократными циклами чтения/записи что пагубно влияет на SSD накопители. Компакт-диски с DRM нагружают оптические приводы, а несанкционированное установка различных прошивок приводит к быстрому износу оборудования.

Следуя выводам можно сказать, что внедрение DRM-защиты в продукт авторского права является нецелесообразным, что подтверждают и другие исследования, связанные с этой темой. Но пока крупные правообладатели с многомиллиардными прибылями не спешат отказываться от этой системы, что привело общество к мировой борьбе против неё. Данное исследование же позволит населению Казахстана войти в эту борьбу и внести свой вклад.

Правообладатели же должны идти навстречу пользователям. Отказываться от DRM, а в регионах с высоким уровнем пиратства устанавливать демократичные цены на продукты. А главное делать их качественными.

Разработчики же новых DRM учтут ошибки прошлых систем, что позволит сделать новые DRM безопасными и ненавязчивыми.

Список использованной литературы

1.Закон Республики Казахстан «Об авторском праве и смежных правах» от 10 июня 1996 года № 6-1 (с изменениями, внесенными Законом РК от 09.07.04 г. № 586-II)

2.Гражданский Кодекс Республики Казахстан от 27.12.1994 г. № 268-XII Алматы 2007г.

3. Бабкин С.А. Интеллектуальная собственность в Интернет. — М.: АО "Центр ЮрИнфоР", 2006. — 512 с.

4. Луцкер А. П. (Арнольд П.). Авторское право в цифровых технологиях и СМИ :Товарные знаки ; Телевидение ; Интернет ; Образование ; Мультимедиа ; Радио = Content rights for creative professionals : Copyrights and trademarks in a digital age / с науч. коммент. А. Серго. — М.: КУДИЦ- ОБРАЗ, 2005. — С. 100-109. — 416 с.

5. Управление правами в области цифровой информации: практическое руководство = Managing didital rights a practitioner's guide / под ред. Поля Педли; пер. с англ. А.И.Земсков; науч. ред. пер. Я. Л. Шрайберг. — М.: Омега-Л, 2010. — 204 с.

6. Крис Касперски, Ева Рокко. Искусство дизассемблирования. — СПб.: БХВ-Петербург, 2008. — 896 с.

7. Владислав Пирогов. Ассемблер для Windows. — СПб.: БХВ-Петербург, 2007. — 896 с.

8. Владислав Пирогов. Ассемблер и дизассемблирование. — СПб.: БХВ-Петербург, 2006. — 464 с.

9. Скляров Д.В. Искусство защиты и взлома информации. — СПб.: БХВ-Петербург, 2004. — 288 с.

10. Панов А.С. Реверсинг и защита программ от взлома — СПб.: БХВ-Петербург, 2006. — 256 с.

12. Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 - Информационные системы – Алматы: АУЭС; 2013. –24 с.

13. Комплексная оценка эффективности мероприятий, направленных на ускорение научно-технического прогресса.Методические рекомендации и комментарии по их применению. -Москва, 2003.

14. Шепеленко Г.И. Экономика, организация и планирование производства на предприятии. Учебное пособие - Ростов-на-дону,«МАРТ», 2004.

15. СанПиН 2.2.2/2.4.1340–03. «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы». – М: Минздрав России, 2003. – 25с.

16. ГОСТ 12.0.003-74 «ССБТ. Опасные и вредные производственные факторы. Классификация». М: Изд-во стандартов, 2004. – 4 с.

17. Безопасность жизнедеятельности. /Под ред. Н.А. Белова - М.: Знание, 2000 - 364с.

18. Самгин Э.Б. Освещение рабочих мест. – М.: МИРЭА, 1989. – 186с.

19. Борьба с шумом на производстве: Справочник / Е.Я. Юдин, Л.А. Борисов; Под общ. ред. Е.Я. Юдина – М.: Машиностроение, 1985. – 400с., ил.

Приложение

Исходники .asm кода

```
[X-CODE INJECT Anti-ISDP]
CPU Disasm
Address   Hex dump          Command
011DB840    64:8B35 1800000 MOV ESI,DWORD PTR FS:[18] //TEB
011DB847    8B76 30         MOV ESI,DWORD PTR DS:[ESI+30] //&TEB.Process database
011DB84A       893E                 MOV DWORD PTR DS:[ESI],EDI TEB.Process database->BeginDebugged = 0 (заодно остальные три флага, которые все равно равны нулю/сброшены)
011DB84C    31F6            XOR ESI,ESI
011DB84E    90              NOP
011DB84F    90              NOP
011DB850    90              NOP
011DB851    90              NOP
011DB852    90              NOP
011DB853    68 77BA1D01     PUSH 011DBA77                          // ASCII "NTDLL"
011DB858                A1      1ABC4C01                MOV      EAX,DWORD      PTR DS:[<&KERNEL32.GetModuleHandleA>]
011DB85D    FFD0            CALL EAX
011DB85F     68 7EBA1D01      PUSH 011DBA7E                                  // ASCII "NtQueryInformationProcess"
011DB864    50              PUSH EAX
011DB865    A1 46BD4C01     MOV EAX,DWORD PTR DS:[<&KERNEL32.GetProcAddress>]
011DB86A    FFD0            CALL EAX
011DB86C    8B35 76BE4C01   MOV ESI,DWORD PTR DS:[<&KERNEL32.VirtualProtect>]
011DB872            68   9ABA1D01            PUSH   011DBA9A   //->Текущий доступ == PAGE_EXECUTE_READ(20h)
011DB877            6A   40                      PUSH   40   //Новый атрибут доступа - PAGE_EXECUTE_READ_WRITE
011DB879    89C7            MOV EDI,EAX
011DB87B    68 00010000     PUSH 100 //100h, чтобы гарантировано изменить доступ
011DB880    50              PUSH EAX
011DB881    FFD6            CALL ESI //меняем атрибуты
011DB883    83C7 06         ADD EDI,6 //сдвигаемся на операнд MOV EDX, [KiFastSystemCall]
011DB886    B8 B7B81D01     MOV EAX,011DB8B7
011DB88B         8707                    XCHG DWORD PTR DS:[EDI],EAX //меняем в (Zw)NtQueryInformationProcess KiFastSystemCall на свой обработчик (выполняем перезапись)
011DB88D    83EF 06         SUB EDI,6 //возвращаемся в начало
011DB890     A3 BBB81D01       MOV DWORD PTR DS:[11DB8BB],EAX //сохраним у себя указатель на KiFastSystemCall
011DB895    66:B8 4000      MOV AX,40 // PAGE_EXECUTE_READ_WRITE
011DB899         66:A3  9ABA1D01    MOV WORD PTR DS:[11DBA9A],AX //возвращаем оригинальные атрибуты доступа
011DB89F    68 9ABA1D01     PUSH 011DBA9A // Предыдущий измененный доступ
011DB8A4    6A 20           PUSH 20 //PAGE_EXECUTE_READ
011DB8A6    68 00010000     PUSH 100
011DB8AB    57              PUSH EDI
```

```
011DB8AC    FFD6           CALL ESI //делаем все как было
011DB8AE    31FF           XOR EDI,EDI
011DB8B0    89FE           MOV ESI,EDI
011DB8B2  ^ E9 79C2FFFF    JMP 011D7B30 //переходим к настоящей EP
011DB8B7    C0B8 1D010000 0
011DB8BE    0000           //АДРЕС НАШЕГО ОБРАБОТЧИКА/СОХР. KiFastSystemCall
011DB8C0    8B4CE4 0C      MOV ECX,DWORD PTR SS:[ESP+0C] //НАШ ОБРАБОТЧИК. Получаем запрашиваемый ProcessInfoClass
011DB8C4    B5 07          MOV CH,7 //ProcessDebugPort
011DB8C6    28CD           SUB CH,CL
011DB8C8    75 0E          JNE SHORT 011DB8D8
011DB8CA    31C0           XOR EAX,EAX //Выполняем дезориентирующий маневр
011DB8CC    8B4CE4 10      MOV ECX,DWORD PTR SS:[ESP+10] //Указатель на  Buffer
011DB8D0    8901           MOV DWORD PTR DS:[ECX],EAX //DebugPort = NULL
011DB8D2    83C4 04        ADD ESP,4
011DB8D5    C2 1400        RETN 14 //Возвращаем управление
011DB8D8    8B0D BBB81D01  MOV ECX,DWORD PTR DS:[11DB8BB] //Для остальных ProcessInfoClass осуществляем переход в KiFastSystemCall
011DB8DE    8B09           MOV ECX,DWORD PTR DS:[ECX]
011DB8E0    51             PUSH ECX
011DB8E1    31C9           XOR ECX,ECX
011DB8E3    C3             RETN //переходим в KiFastSystemCall
[BINARY]
68 77 BA 1D 01 A1 1A BC 4C 01 FF D0 68 7E BA 1D
01 50 A1 46 BD 4C 01 FF D0 8B 35 76 BE 4C 01 68
9A BA 1D 01 6A 40 89 C7 68 00 01 00 00 50 FF D6
83 C7 06 B8 B7 B8 1D 01 87 07 83 EF 06 A3 BB B8
1D 01 66 B8 40 00 66 A3 9A BA 1D 01 68 9A BA 1D
01 6A 20 68 00 01 00 00 57 FF D6 31 FF 89 FE E9
79 C2 FF FF C0 B8 1D 01 00 00 00 00 00 8B 4C E4
0C B5 07 28 CD 75 0E 31 C0 8B 4C E4 10 89 01 83
C4 04 C2 14 00 8B 0D BB B8 1D 01 8B 09 51 31 C9
C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 4E 54 44 4C 4C 00 00 4E 74 51 75 65
72 79 49 6E 66 6F 72 6D 61 74 69 6F 6E 50 72 6F
63 65 73 73 00
```

[X-CODE INJECT Ist part]
[CPU Disasm]
Address Hex dump Command

```
011DB27A  |.  0FB659 06     MOVZX EBX,BYTE PTR DS:[ECX+6]
011DB27E  |.  E9 7D040000   JMP 011DB700 // переход на первую часть X-кода
011DB283  |   90           NOP
011DB284  |   90           NOP
011DB285  |   90           NOP
011DB286  |>  03FE          ADD EDI,ESI // адрес возврата

011DB700  |> \01F7         ADD EDI,ESI // 1я эмулируемая инструкция
011DB702  |.  01DE         ADD ESI,EBX // 2я эмулируемая инструкция
011DB704  |.  0FB659 07     MOVZX EBX,BYTE PTR DS:[ECX+7] // 3я эмулируемая инструкция
011DB708  |.  83F8 6C       CMP EAX,6C // Чтобы минимизировать количество перезаписей верхушки GetDriveTypeA, введем условие
011DB70B  |.^ 0F85 75FBFFFF JNE 011DB286 // Возврат, если не 6Ch
011DB711  \.  E9 83000000   JMP 011DB799 // Переход на X-код

011DB799  /> \60           PUSHAD //сохраняем текущие регистры в стек
011DB79A  |.  68 B6B71D01   PUSH cnc3game.011DB7B6; /ModuleName = "KERNEL32"
011DB79F  |.  FF15 1ABC4C01 CALL DWORD PTR DS:[<&KERNEL32.GetModuleH ; \KERNEL32.GetModuleHandleA
011DB7A5  |.  68 C3B71D01   PUSH cnc3game.011DB7C3; /Procname = "GetDriveTypeA"
011DB7AA  |.  50           PUSH EAX; |hModule
011DB7AB  |.  FF15 46BD4C01 CALL DWORD PTR DS:[<&KERNEL32.GetProcAdd ; \KERNEL32.GetProcAddress
011DB7B1  |.  EB 1E         JMP SHORT 011DB7D1 // в EAX адрес GetDriveTypeA
011DB7B3  |   90           NOP
011DB7B4  |   90           NOP
011DB7B5  |   90           NOP
011DB7B6  |.  4B 45 52 4E 4 ASCII "KERNEL32",0
011DB7BF  |   00           DB 00
011DB7C0  |   00           DB 00
011DB7C1  |   00           DB 00
011DB7C2  |   00           DB 00
011DB7C3  |.  47 65 74 44 7 ASCII "GetDriveTypeA",0
011DB7D1  |>  89C1          MOV ECX,EAX // скопируем в ECX адрес из EAX
```

011DB7D3 |. 2D E3B61D01 SUB EAX,cnc3game.011DB6E3 // расстояние от GDTA до первой инструкции второй части X-кода
011DB7D8 |. BF FBFFFFFF MOV EDI,-5 // EDI = FFFFFFFBh
011DB7DD |. 29C7 SUB EDI,EAX // получаем искомые байты для операнда
011DB7DF |. C601 E9 MOV BYTE PTR DS:[ECX],0E9 // ставим в начало GDTA опкод JMP. Если секция text библиотеки kernel32 не имеет атрибута Write, на этой команде возникнет исключение access_violation
011DB7E2 |. 8979 01 MOV DWORD PTR DS:[ECX+1],EDI // пишем операнд. Обратите внимание, что операнд запишется наоборот, что нам и надо!
011DB7E5 |. 61 POPAD // восстановим регистры
011DB7E6 \.^ E9 9BFAFFFF JMP 011DB286 // возвращаемся

[BINARY]
#INCLUDE TO [BINARY_ALL]

[X-CODE INJECT 2nd part]
[CPU Disasm]
Address Hex dump Command Comments
011DB6E3 . 58 POP EAX // ВЫТАСКИВАЕМ АДРЕС ВОЗВРАТА. EAX в расход.
011DB6E4 . A3 EBB61D01 MOV DWORD PTR DS:[11DB6EB],EAX // Сохраняем его у себя
011DB6E9 . EB 31 JMP SHORT 011DB71C

011DB71C > \50 PUSH EAX //закидываем обратно
// PUSH XXXXXXXX – казалось бы было правильным и тривиальным решением закинуть в стек адрес возврата на свой обработчик. Однако это не так - "съедут" все указатели аргументов! Даже если стек выровнять, то эмулируемая ниже PUSH EBP затрет оригинальный адрес возврата.
Есть вариант - сохранить EBP в EDI и присвоить первому адрес оригинального возврата. Однако это дикий способ. Если X-код не приведет все в нормальный вид, то возникнет большой переполох. Не будем забивать голову, самый оптимальный вариант - просто подкорректировать оригинальный адрес, предварительно сохранив его, а после в X-коде выполнить переход на него.
011DB71D . EB 08 JMP SHORT 011DB727

011DB727 > \60 PUSHAD
011DB728 . 31C0 XOR EAX,EAX
011DB72A . A0 EBB61D01 MOV AL,BYTE PTR DS:[11DB6EB] //последний байт адреса возврата
011DB72F . BE EBB61D01 MOV ESI,cnc3game.011DB6EB // адрес последнего байта, для декремента
011DB734 . BF 00B91D01 MOV EDI,cnc3game.011DB900 // адрес для записи готовых ASCII кодов
011DB739 . 31C9 XOR ECX,ECX
011DB73B . B5 30 MOV CH,30 // первичный ACSII код для числа ноль, старший разряд байта DL
011DB73D . B1 30 MOV CL,30 // первичный ACSII код для числа ноль, младший разряд байта DL
011DB73F . 31D2 XOR EDX,EDX // первичное значение эталонного числа - ноль
011DB741 > 38D0 CMP AL,DL // равно ли сравниваемое и эталонное?
011DB743 . 74 21 JE SHORT 011DB766 // Равно!

```
011DB745   . EB 00        JMP SHORT 011DB747 // Не равно!
011DB747   > 80F9 39       CMP CL,39 // В младшем разряде число 9h?
011DB74A   . 75 03         JNE SHORT 011DB74F
011DB74C   . B1 40         MOV CL,40 // Да! Правим на 40h! Иначе – пропускаем.
011DB74E   . 90            NOP
011DB74F   > 80F9 46       CMP CL,46 // В младшем разряде число Fh?
011DB752   . 74 09         JE SHORT 011DB75D
011DB754   . 90            NOP
011DB755   . 90            NOP
011DB756   . 90            NOP
011DB757   . FEC1          INC CL // Нет! Увеличиваем младший разряд и эталонный байт
011DB759   . FEC2          INC DL
011DB75B   .^ EB E4        JMP SHORT 011DB741 // Сравниваем
011DB75D   > 80E9 16       SUB CL,16 // Да! 46h – 16h = 30h. Ноль в младшем разряде.
011DB760   . EB 28         JMP SHORT 011DB78A
011DB762     90            NOP
011DB763     90            NOP
011DB764     90            NOP
011DB765     90            NOP
011DB766   > 882F          MOV BYTE PTR DS:[EDI],CH //AL=DL! Пишем старший ACSII код
числа
011DB768   . 47            INC EDI
011DB769   . 880F          MOV BYTE PTR DS:[EDI],CL //AL=DL! Пишем младший ACSII код
числа
011DB76B   . 47            INC EDI
011DB76C   . 46            INC ESI
011DB76D   . 81FE EFB61D01 CMP ESI,cnc3game.011DB6EF // Условие выхода из цикла
011DB773   . 75 0A         JNE SHORT 011DB77F //Если не выполняется, то …
011DB775   . B8 00B91D01   MOV EAX,cnc3game.011DB900 // Конвертированная ACSII
строка
011DB77A   .^ E9 51FFFFFF  JMP 011DB6D0 // MessageBoxA
011DB77F   > 90            NOP
011DB780   . B5 30         MOV CH,30 // … начинаем новый заход
011DB782   . B1 30         MOV CL,30
011DB784   . 31D2          XOR EDX,EDX
011DB786   . 8A06          MOV AL,BYTE PTR DS:[ESI]
011DB788   .^ EB B7        JMP SHORT 011DB741
011DB78A   > 80FD 39       CMP CH,39 // В старшем разряде число 9h?
011DB78D   . 75 04         JNE SHORT 011DB793
011DB78F   . B5 40         MOV CH,40
011DB791   . 90            NOP
011DB792   . 90            NOP
011DB793   > FEC2          INC DL
011DB795   . FEC5          INC CH
011DB797   .^ EB A8        JMP SHORT 011DB741 // Возврат к проверке

011DB6D0   > /60           PUSHAD
011DB6D1   . |6A 40        PUSH 40                          ; /Type =
MB_OK|MB_ICONASTERISK|MB_DEFBUTTON1|MB_APPLMODAL
011DB6D3   . |50           PUSH EAX                         ; |Caption
011DB6D4   . |90           NOP                              ; |
```

```
011DB6D5   . |6A 00        PUSH 0                          ; |Text = NULL
011DB6D7   . |6A 00        PUSH 0                          ; |hOwner = NULL
011DB6D9     . |FF15 FEB94C01 CALL DWORD PTR DS:[<&USER32.MessageBoxA> ; \USER32.MessageBoxA // Показ адреса возврата
011DB6DF   . |EB 35        JMP SHORT 011DB716

011DB716   > \61          POPAD
011DB717   . E9 CF000000   JMP 011DB7EB // последний этап-возвращение в kernel32

011DB7EB   > \68 B6B71D01    PUSH cnc3game.011DB7B6                ; /ModuleName = "KERNEL32"
011DB7F0     . FF15 1ABC4C01 CALL DWORD PTR DS:[<&KERNEL32.GetModuleH ; \KERNEL32.GetModuleHandleA
011DB7F6   . 68 C3B71D01    PUSH cnc3game.011DB7C3                ; /Procname = "GetDriveTypeA"
011DB7FB   . 50          PUSH EAX                          ; |hModule
011DB7FC     . FF15 46BD4C01 CALL DWORD PTR DS:[<&KERNEL32.GetProcAdd ; \KERNEL32.GetProcAddress
// сейчас по адресу 011DB825 будет записан прыжок обратно в kernel32. Так как после POPAD X-код не имеет больше права манипулировать с регистрами, мы позаботимся о прыжке заранее.
011DB802   . 2D 25B81D01    SUB EAX,cnc3game.011DB825 // количество байт, которые потребуется перепрыгнуть.
// смотри! По идеи мы ведь должны отнять от EAX сейчас 5 байт длинной инструкции JMP(которая будет по адресу 011DB825). Однако, в таком случае мы попадем в начало GetDriveTypeA, где стоит наш переход во вторую часть X-кода, который также занимает 5 байт! Значит, после мы должны накинуть его 5 байт. Выходит: EAX - 5 + 5 = EAX. Отсюда мораль – никаких дополнительных действий не требуется. Мы попадем точно по месту назначения.
011DB807   . C605 25B81D01 MOV BYTE PTR DS:[11DB825],0E9 //переход
011DB80E     . A3 26B81D01    MOV DWORD PTR DS:[11DB826],EAX // его операнд (DWORD = 4 байта)
011DB813   . C605 A043FB00 MOV BYTE PTR DS:[0FB43A0],74 // не обращайте внимания!
011DB81A   . C605 A63FFB00 MOV BYTE PTR DS:[0FB3FA6],0F9 // эт я какой-то переход правил в защите
011DB821   . 61          POPAD // оригинальные регистры в студию
// эмулируем первые три перекрытые инструкции в API. MOV EDI, EDI не несет в себе никакого полезного смысла, отсюда мораль – эмулируем только открытие кадра стека
011DB822   . 55          PUSH EBP
011DB823   . 89E5        MOV EBP,ESP
011DB825   . 0000          ADD BYTE PTR DS:[EAX],AL // когда сюда докатится EIP, здесь будет переход в kernel32 (оригинальный адрес GetDriveTypeA со смещением в 5 байт)
011DB827   . 0000        ADD BYTE PTR DS:[EAX],AL
011DB829   . 0000        ADD BYTE PTR DS:[EAX],AL
011DB82B   . 0000        ADD BYTE PTR DS:[EAX],AL
011DB82D   . 0000        ADD BYTE PTR DS:[EAX],AL

[BINARY_ALL]
60 6A 40 50 90 6A 00 6A 00 90 FF 15 FE B9 4C 01
EB 34 00 58 A3 EB B6 1D 01 EB 31 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
01 F7 01 DE 0F B6 59 07 83 F8 6C 0F 85 75 FB FF
FF E9 83 00 00 00 61 E9 CF 00 00 00 50 EB 08 90
90 90 68 C2 B7 1D 01 60 33 C0 A0 EB B6 1D 01 BE
EB B6 1D 01 BF 00 B9 1D 01 33 C9 B5 30 B1 30 33
D2 3A C2 74 21 EB 00 80 F9 39 75 03 B1 40 90 80
F9 46 74 09 90 90 90 FE C1 FE C2 EB E4 80 E9 16
EB 28 90 90 90 90 88 2F 47 88 0F 47 46 81 FE EF
B6 1D 01 75 0A B8 00 B9 1D 01 E9 51 FF FF FF 90
B5 30 B1 30 33 D2 8A 06 EB B7 80 FD 39 75 04 B5
40 90 90 FE C2 FE C5 EB A8 60 68 B6 B7 1D 01 FF
15 1A BC 4C 01 68 C2 B7 1D 01 50 FF 15 46 BD 4C
01 EB 1E 90 90 90 4B 45 52 4E 45 4C 33 32 00 00
00 00 47 65 74 44 72 69 76 65 54 79 70 65 41 00
00 89 C1 2D E3 B6 1D 01 BF FB FF FF FF 29 C7 C6
01 E9 89 79 01 61 E9 9B FA FF FF 68 B6 B7 1D 01
FF 15 1A BC 4C 01 68 C2 B7 1D 01 50 FF 15 46 BD
4C 01 83 E8 00 2D 28 B8 1D 01 C6 05 28 B8 1D 01
E9 A3 29 B8 1D 01 C6 05 16 B8 1D 01 C6 C6 05 16
B8 1D 01 C6 61 55 89 E5 00 00 00 00 00 00 00 00
```

[X-CODE INJECT IIIrd part]
[CPU Disasm]

```
011DBEE8   60            PUSHAD
011DBEE9   BE 00104000   MOV ESI,cnc3game.00401000 //начало секции .text
011DBEEE   BF 00005000   MOV EDI,cnc3game.00500000 //РЕЗЕРВ!
011DBEF3   31C0          XOR EAX,EAX
011DBEF5   803E FF       CMP BYTE PTR DS:[ESI],0FF //проверям на первый байт инструкции JMP DWORD PTR DS:[adress]
011DBEF8   74 03         JE SHORT cnc3game.011DBEFD
011DBEFA   46            INC ESI
011DBEFB   ^EB F8        JMP SHORT cnc3game.011DBEF5 //если не FF то переходим на следующий адрес (инкрементом)
011DBEFD   807E 01 25    CMP BYTE PTR DS:[ESI+1],25  //если FF то на один адрес выше должен быть байт 25
011DBF01   74 03         JE SHORT cnc3game.011DBF06
011DBF03   46            INC ESI
011DBF04   ^EB EF        JMP SHORT cnc3game.011DBEF5   //если не 25 то переходим на следующий адрес (инкрементом)
011DBF06   8B4E 02       MOV ECX,DWORD PTR DS:[ESI+2]  //нашли JMP DWORD PTR DS:[adress]. В ECX операнд adress
011DBF09   8B39          MOV EDI,DWORD PTR DS:[ECX]    //читаем по нему заложенный операнд и сохраняем его в EDI
011DBF0B   2BFE          SUB EDI,ESI                   //начинаем рассчитывать байты операнда уже для нашего прямого прыжка
011DBF0D   83EF 05       SUB EDI,5                     //корректируем на длинну инструкции прыжка
011DBF10   C606 E9       MOV BYTE PTR DS:[ESI],0E9     //пишем новый опкод прыжка
011DBF13   897E 01       MOV DWORD PTR DS:[ESI+1],EDI  //пишем байты рассчитанного операнда
011DBF16   C646 05 90    MOV BYTE PTR DS:[ESI+5],90    //NOP'им оставшейся байт от 6-ти байтной JMP DWORD PTR DS:[adress]
```

```
011DBF1A   81FE D342A200    CMP ESI,cnc3game.00A242D3     //достигнут ли конец секции .text ?
011DBF20   46               INC ESI
011DBF21  ^7C D2            JL SHORT cnc3game.011DBEF5    //если нет, цикл повторяется
011DBF23   61               POPAD                         //иначе, работа окончена
```

```
[BINARY]
60 BE 00 10 40 00 BF 00 00 50 00 31 C0 80 3E FF 74 03 46 EB F8 80 7E 01 25 74 03 46 EB EF 8B 4E
02 8B 39 2B FE 83 EF 05 C6 06 E9 89 7E 01 C6 46 05 90 81 FE D3 42 A2 00 46 7C D2 61
```

```
60 BE 00 10 40 00 BF 00 00 50 00 31 C0 80 3E FF 74 03 46 EB F8 80 7E 01 25 74 03 46 EB EF 8B 4E
02 81 F9 00 00 5D 01 72 03 46 EB E1 8B 39 81 FF 00 00 5D 01 7C 03 46 EB D4 2B FE 83 EF 05 C6 06
E9 89 7E 01 C6 46 05 90 81 FE D3 42 A2 00 46 40 7F BB 61 00 00 00 00
```

```
00A242EC      . C705 D342A200 >MOV DWORD PTR DS:[A242D3],NODVD.00C2D020 //точка отсчета для копирования адресов от "диспетчера по стрелкам"
00A242F6      . C705 D742A200 >MOV DWORD PTR DS:[A242D7],NODVD.00C2D5C0 //точка отсчета для копирования адресов от "mem-ремотника"
00A24300      . C705 DB42A200 >MOV DWORD PTR DS:[A242DB],NODVD.00C2D910 //точка отсчета для копирования адресов от "конечной"
00A2430A    . 8B35 D342A200  MOV ESI,DWORD PTR DS:[A242D3]              //адрес назначения граббера для "диспетчера по стрелкам". берем следующий отсчитываемый адрес в ESI
00A24310    . 8916           MOV DWORD PTR DS:[ESI],EDX                 //POP DWORD PTR DS:[EDX], копируем место "стрелки"
00A24312    . 52             PUSH EDX                                   //сохраним EDX в стеке для собственных нужд
00A24313    . 3E:8B5424 04   MOV EDX,DWORD PTR DS:[ESP+4]               //POP DWORD PTR DS:[EDX], копируем из стека значение "путь конечного назначения"
00A24318    . 8956 04        MOV DWORD PTR DS:[ESI+4],EDX               //копируем себе "путь конечного назначения", при этом смещаемся на 4 байта(т.к. было скопировано ранее место "стрелки")
00A2431B    . 5A             POP EDX                                    //восстанавливаем EDX
00A2431C    . 8305 D342A200 >ADD DWORD PTR DS:[A242D3],8                //смещаем на 8 байт точку отсчета для следующих значений (переходим на свободное место)
00A24323    . FF05 DF42A200  INC DWORD PTR DS:[A242DF]                  //статистика. счетчик скопированных значений
00A24329    . 8F02           POP DWORD PTR DS:[EDX]                     //эмулируемая команда
00A2432B    . 89DF           MOV EDI,EBX                                //эмулируемая команда
00A2432D    . 83C7 04        ADD EDI,4                                  //эмулируемая команда
00A24330    .-E9 52B01102    JMP NODVD.02B3F387                         //возврат
00A24335    . 8B35 D742A200  MOV ESI,DWORD PTR DS:[A242D7]              //адрес назначения граббера для "mem-ремотника". берем следующий отсчитываемый адрес в ESI
00A2433B    . 8916           MOV DWORD PTR DS:[ESI],EDX
00A2433D    . 52             PUSH EDX
```

```
00A2433E   . 3E:8B5424 04   MOV EDX,DWORD PTR DS:[ESP+4]
00A24343   . 8956 04        MOV DWORD PTR DS:[ESI+4],EDX
00A24346   . 5A             POP EDX
00A24347   . 8305 D742A200 >ADD DWORD PTR DS:[A242D7],8
00A2434E   . FF05 E342A200  INC DWORD PTR DS:[A242E3]
00A24354   . 8F02           POP DWORD PTR DS:[EDX]
00A24356   . 8B1424         MOV EDX,DWORD PTR SS:[ESP]
00A24359   .-E9 9A4D6001    JMP NODVD.020290F8
00A2435E   . 8D6424 30      LEA ESP,DWORD PTR SS:[ESP+30]
00A24362   . 57             PUSH EDI
00A24363   . 56             PUSH ESI
00A24364    . 8B35 DB42A200  MOV ESI,DWORD PTR DS:[A242DB]                    //адрес
назначения граббера для "конечной". берем следующий отсчитываемый адрес в ESI
00A2436A   . 3E:8B3CE4      MOV EDI,DWORD PTR DS:[ESP]
00A2436E   . 893E           MOV DWORD PTR DS:[ESI],EDI
00A24370   . 3E:8B7CE4 08   MOV EDI,DWORD PTR DS:[ESP+8]
00A24375   . 90             NOP
00A24376   . 90             NOP
00A24377   . 90             NOP
00A24378   . 897E 04        MOV DWORD PTR DS:[ESI+4],EDI
00A2437B   . 8305 DB42A200 >ADD DWORD PTR DS:[A242DB],8
00A24382   . FF05 E742A200  INC DWORD PTR DS:[A242E7]
00A24388   . 5E             POP ESI
00A24389   . 5F             POP EDI
00A2438A   . C2 0400        RETN 4
00A2438D     00             DB 00
```

```
C7 05 D3 42 A2 00 20 D0 C2 00 C7 05 D7 42 A2 00 C0 D5 C2 00 C7 05 DB 42 A2 00 10 D9 C2
00 8B 35
D3 42 A2 00 89 16 52 3E 8B 54 24 04 89 56 04 5A 83 05 D3 42 A2 00 08 FF 05 DF 42 A2 00 8F
02 89
DF 83 C7 04 E9 52 B0 11 02 8B 35 D7 42 A2 00 89 16 52 3E 8B 54 24 04 89 56 04 5A 83 05 D7
42 A2
00 08 FF 05 E3 42 A2 00 8F 02 8B 14 24 E9 9A 4D 60 01 8D 64 24 30 57 56 8B 35 DB 42 A2 00
3E 8B
3C E4 89 3E 3E 8B 7C E4 08 90 90 90 89 7E 04 83 05 DB 42 A2 00 08 FF 05 E7 42 A2 00 5E 5F
C2 04
00 00
```

[X-CODE INJECT IIIrd part]
Учитывается, что прыжок совершается после:
ADD EBX,24
MOV DWORD PTR DS:[EBX],0

DWORD [011DB90A] = 00C2D000 (.tls)

```
011DB90F   90               NOP
011DB910   90               NOP
011DB911   90               NOP
011DB912    83C3 0C                ADD EBX,0C //EBX – НАЧАЛО ОБЛАСТИ ЯЧЕЕК ДЛЯ
ХРАНЕНИЯ АДРЕСОВ/ЗНАЧЕНИЙ ОСТРОВКАМИ
```

011DB915 8BF3 MOV ESI,EBX //ПОРТИМ В ESI
011DB917 81C3 F4030000 ADD EBX,3F4 //ОКОНЧАЕНИЕ ОБЛАСТИ ЯЧЕЕК В ГЛАВНОМ ХРАНИЛИЩЕ (СЛЕДУЯ ЭТИКЕТУ НЕОБХОДИМО ПРЕДУСМОТРЕТЬ ВЫХОД ИЗ ЦИКЛА, ЕСЛИ АДРЕСА НЕ ОКАЖЕТСЯ В ЯЧЕЙКАХ ВООБЩЕ, НО ЭТО УСЛОВИЕ НИКОГДА НЕ БУДЕТ ВЫПОЛНЯТСЯ – АДРЕС ПО ЛЮБОМУ БУДЕТ В ГЛАВНОМ ХРАНИЛИЩЕ, ИНАЧЕ VM ПРОСТО НЕПРАВИЛЬНО РАБОТАЕТ!)
011DB91D 8BFB MOV EDI,EBX //ПОРТИМ В EDI
011DB91F 8B4C24 24 MOV ECX,DWORD PTR SS:[ESP+24] //ДОСТАЕМ АДРЕС ЗАПРАШИВАЕМОЙ ФУНКЦИИ
011DB923 BA 0AB91D01 MOV EDX,cnc3game.011DB90A //АДРЕС, КУДА ГРАББЕР БУДЕТ СКЛАДЫВАТЬ ПОЛУЧЕННЫЕ АДРЕСА
011DB928 8B12 MOV EDX,DWORD PTR DS:[EDX] //ЧИТАЕМ ТЕКУЩИЙ УКАЗАТЕЛЬ НА ОБЛАСТЬ
011DB92A 8B1E MOV EBX,DWORD PTR DS:[ESI] //ЧИТАЕМ DWORD В ЯЧЕЙКЕ
011DB92C 85DB TEST EBX,EBX //ПРОВЕРЯЕМ НА НОЛЬ(СИТУАЦИЯ КОГДА ПУСТАЯ ЯЧЕЙКА)
011DB92E 74 06 JE SHORT cnc3game.011DB936
011DB930 8B03 MOV EAX,DWORD PTR DS:[EBX] //ЧИТАЕМ ДВОЙНОЕ СЛОВО ПО ПРЕДПОЛАГАЕМОМУ АДРЕСУ
011DB932 3BC1 CMP EAX,ECX
011DB934 74 05 JE SHORT cnc3game.011DB93B
011DB936 83C6 04 ADD ESI,4 //СЛЕДУЩАЯ ЯЧЕЙКА
011DB939 ^EB F3 JMP SHORT cnc3game.011DB9A
//ОК! НАШЛИ ЯЧЕЙКУ, ПО КОТОРОЙ БЫЛА ПЕРЕВЕДЕНА СТРЕЛКА//
011DB93B 891A MOV DWORD PTR DS:[EDX],EBX //ПИШЕМ АДРЕС ИЗ ЯЧЕЙКИ (ОПЕРАНД СТРЕЛКИ)
011DB93D 83C2 04 ADD EDX,4 //СМЕЩАЕМСЯ
011DB940 8902 MOV DWORD PTR DS:[EDX],EAX //ПИШЕМ АДРЕС ЗАПРАШИВАЕМОЙ ПРОЦЕДУРЫ
011DB942 B9 0AB91D01 MOV ECX,cnc3game.011DB90A // АДРЕС, КУДА ГРАББЕР БУДЕТ СКЛАДЫВАТЬ ПОЛУЧЕННЫЕ АДРЕСА
011DB947 83C2 04 ADD EDX,4 //СМЕЩАЕМСЯ В СЛЕДУЮЩУЮ СВОБОДНУЮ ОБЛАСТЬ
011DB94A 8911 MOV DWORD PTR DS:[ECX],EDX //КОРРЕКТИРУЕМ У СЕБЯ УКАЗАТЕЛЬ
//ПРОЦЕДУРА ВЫХОДА ИЗ VM //
011DB94C 9D POPFD //ВЫТАСКИВАЕМ ФЛАГИ
011DB94D 61 POPAD //ВЫТАСКИВАЕМ РЕГИСТРЫ
011DB94E 8D6424 24 LEA ESP,DWORD PTR SS:[ESP+24] //КОРРЕТИРУЕМ УКАЗАТЕЛЬ
011DB952 C2 0400 RETN 4 //ОСУЩЕСТВЛЯЕМ ПЕРЕХОД В ЗАПРАШИВАЕМУЮ ПРОЦЕДУРУ И “СНИМАЕМ” АРГУМЕНТЫ ИЗ БАЗЫ ЗАПРОСОВ

[BINARY]
90 90 90 83 C3 0C 8B F3 81 C3 F4 03 00 00 8B FB 8B 4C 24 24 BA 0A B9 1D 01 8B 12 8B 1E 85 DB 74
06 8B 03 3B C1 74 05 83 C6 04 EB EF 89 1A 83 C2 04 89 02 B9 0A B9 1D 01 83 C2 04 89 11 9D 61 8D
64 24 24 C2 04 00

402F1D – GLOBAL FREE

```
00C93AE3   E8 58140B00      CALL cnc3game.00D44F40

011DB6D0    BE 50D3BC00     MOV ESI,OFFSET 00BCD350
011DB6D5    89F7            MOV EDI,ESI
011DB6D7    83C7 04         ADD EDI,4
011DB6DA    8B06            MOV EAX,DWORD PTR DS:[ESI]
011DB6DC    85C0            TEST EAX,EAX
011DB6DE    74 0C           JE SHORT 011DB6EC
011DB6E0    8B0F            MOV ECX,DWORD PTR DS:[EDI]
011DB6E2    8908            MOV DWORD PTR DS:[EAX],ECX
011DB6E4    83C7 08         ADD EDI,8
011DB6E7    83C6 08         ADD ESI,8
011DB6EA  ^ EB EE           JMP SHORT 011DB6DA
011DB6EC    90              NOP
011DB6ED    90              NOP
BE 50 D3 BC 00 89 F7 83 C7 04 8B 06 85 C0 74 0C
8B 0F 89 08 83 C7 08 83 C6 08 EB EE 90 90
```